Islander Augmented Reality

Presented to:

Dr. Ahmed Mahdy

Islander Augmented Reality Outline

I Project Overview and Objectives

II Requirements
- i Plain-English Description
- ii Functional Requirements
- iii Non-functional Requirements
- iv Structured Requirements
- v Islander Ambassador Interview
- vi Client proposal meeting

III Analysis
- i Use Case Scenarios
  - View Main Menu
  - Launch IslanderAR View
  - View Options
  - Search Points of Interest
- ii Use Case Realization

IV Design
- i Initial Design Considerations
  - Architecture of the Android Operating System
  - Android activity Life Cycle diagram
  - Activity State diagram
- ii Design Rationale and Decisions
- iii Class Enumeration and Specific Design Detail
- iv Design Diagrams
  - Class Diagram
  - Stereotype Class Diagram
  - Collaboration Diagrams
  - Sequence Diagram
- iii Database
  - Database Design Rationale
  - Crows Foot Entity-Relationship Model
  - Data Dictionary

V Implementation

VI Major Components of the Android Operating System

VII Testing Activities

VIII Appendices
- i Meeting Minutes
- ii Status Reports
- iii User Manual
- iv Source Code Installation Guide
- v Known Issues
- vi Future Work
- vii Bibliography

Project Overview and Objectives

IslanderAR is an augmented reality (AR) application designed to run on cellular smart phones. Augmented reality is a phrase that describes the merging of a view of the real, physical world with computer generated images. The computer generated images enhance, or augment the reality view, giving more information than either system alone.

IslanderAR is designed to run on the Android™ Mobile Operating system. It allows the user to conduct augmented tours of the Texas A&M Corpus Christi campus. The application works by transmitting the smart phone's camera view to the screen and overlaying icons representing specific campus sites known as Points of Interest, or POIs. As the camera is turned towards a POI, the icon for that POI floats into view in the same position as the POI.

The user can configure how the POIs are displayed based on distance and type, and the user can search for specific POIs using common attributes of the POI such as its name or nickname.

The objectives of this project were to develop an application on a smart phone. Other objectives included learning the new development platform, learning a whole new programming style, learning resource management on a device with limited resources, and experience using a versioning system in a real-world setting.

We chose to develop this application for the above reasons as well as being able to work with emerging technology early in it's lifecycle.

Requirements

Plain-English Description

IslanderAR is an augmented reality (AR) application designed to run on cellular smart phones. Augmented reality is a phrase that describes the merging of a view of the real, physical world with computer generated images. The computer generated images enhance, or augment the reality view, giving more information than either system alone.

IslanderAR is designed to run on the Android™ Mobile Operating system. It allows the user to conduct augmented tours of the Texas A&M University – Corpus Christi campus. The application works by transmitting the smart phone's camera view to the screen and overlaying icons representing specific campus sites known as Points of Interest, or POIs. As the camera is turned towards a POI, the icon for that POI floats into view in the same position as the POI. The POI icon will be displayed even if the actual site is hidden from view, such as behind another structure.

The user can configure how the POIs are displayed based on distance and type, and the user can search for specific POIs using common attributes of the POI such as its name or nickname.

Further details can be found in the IslanderAR User Guide.

Requirements - Functional

1. Functional Requirements (Input)
    1.1. Dynamic axis sensor polling
    1.2. Dynamic GPS polling
    1.3. User selections
    1.4. POI data
    1.5. Web access
2. Functional Requirements (Behavior)
    2.1. Dynamic position calculations
    2.2. Relative POI calculations
    2.3. Uninterrupted display
    2.4. POI information
    2.5. Application menus
    2.6. Device buttons
3. Functional Requirements (Output)
    3.1. POI icons merged with camera view
    3.2. Multiple POI identifiers
    3.3. Current bearing

Current viewing distanc

Requirements - Functional

1. Non-Functional Requirements
   1.1. Splash or welcome screen
   1.2. Simple and easy menu
   1.3. Points of Interest aesthetically displayed
   1.4. POI must provide additional information
      1.4.1. Small text box with basic information
      1.4.2. Custom or existing web page

Requirements - Structural

1. The application should present the user with a welcome screen (or greeter).
2. The user should be able to navigate to the guidance screen.
   2.A. The user should be able to navigate to a basic video tutorial.
      2.A.1. Different language videos?
      2.A.2. Offer the videos on an external site, such as youtube?
      2.A.3. Maintain the video locally or on a school site?
   2.B. The user should be able to access an online guide.
      2.B.1. The guide should provide basic, high-level guidance.
      2.B.2. Again, different language options?
      2.B.3. Guide storage location (plain text makes for small, local files)?
3. The user should be able to navigate to the application using the default settings.
   3.A. The user should see the camera view as if they were looking through the device.
   3.B. The view should have points of interest overlaid on the camera view.
   3.C. The user should be able to move the device, and have the POIs appear on the screen in relation to their physical location.
      3.C.1. As the device is rotated, POIs will come into view and leave the view.
   3.D. The user should be able to click on a POI and receive additional information about it.
   3.E. The application should have the ability to guide the user to a desired POI.
4. The user should be able to filter the POI information displayed.
   4.A. Instead of the default settings, the user should be able to choose the settings they wish to see.
      4.A.1. The user should able to filter for eating facilities.
      4.A.2. The user should able to filter for simple building information.
      4.A.3. The user should able to filter for student services.
      4.A.4. The user should able to filter for restroom/break areas.
5. The user should be able to access a web page for more specific information about a particular POI from the POI icon

Islander Ambassador Interview

Team meeting: 01 October, 2009
Attendees:
Ryan Egger
Patrick Krepps
Marco Longoria
Patrick O'Neal

Guest:
Laura Villalobos

The team met with Laura Villalobos this afternoon. Laura is a former Islander Ambassador (IA), and she was more than willing to take our questions and to provide information.

To begin with, Laura offered alternative contacts as sources of information. She suggested we try to contact Russell Wagner, Sara Hubert, or Nicole Fuentes, all Admissions Counselors in the Office of New Student Recruitment.

Islander tour groups can usually be broken into three groups:
Families, with students interested in the University
High School student groups, with students interested in a day away from their school
Groups of elementary students, with the focus of those tours geared towards generating interest in attending college.

Every tour is required to discuss and show the location of the Learning and Tutoring center. Starting points of the tour vary, based on the group, and based on the IA. The University Services Center is often the starting point. Tours are not generally conducted in a specific order. The tours consist of discussing the various campus institutions and highlights of each.

Some of the things discussed by the IA's are:
Past names of the university
Islander traditions (such as the Islander Revue and the Islander Tribute)
Islander colors and Islander mascot

Common questions include:
Application questions - how to apply, where to apply, how much does it cost?
Admissions requirements?
Questions about how SAT/ACT scores are used
THEA questions
Questions about applying for and receiving financial aid
Questions concerning textbooks.
Living arrangements, such as how room-mates are assigned, community or private bathrooms.
Questions about the campus library
Lots of questions about the Garcia Plaza, especially if there are veterans in the group.
Where is the health center?
How do I register?

Who is my academic advisor?
Where are the offices of campus organizations, such as sororities and fraternities?

#1 building asked about the most is the gym. This is because parents are concerned about their students having enough on-campus activities to choose from so they will not want to leave the campus. Parents also ask for CC RTA schedules and information.
#2 building is the UC. Questions include wanting to know where to purchase Islander souvenirs, where are books purchased, are there off-campus book stores, where is the bank and/or ATM located, and where can I go to eat?

Groups can request bi-lingual tours in both Spanish and English. There are no additional provisions for other ESL students that Laura is aware of.

Other miscellaneous information that is included is unusual facts, such as the first performer to perform at the PAC was Freddy Fender.

Client Proposal Meeting

23 September 2009
Client Proposal Meeting

Attendees:

Leticia Bazan - Client
Patrick Krepps
Patrick O'Neal

Summary:

At 1000 hours, Patrick Krepps & Patrick O'Neal met with Leticia Bazan, Director for the Office of Student Recruitment & New Student Programs. There they proposed to her the teams' idea for an augmented reality smart phone application that will aid students seeking information about various campus resources. Mr. Krepps and Mr. O'Neal informed Ms. Bazan of what the AR-Team would like to accomplish, a proof of concept application designed for open development platform. They also informed her that if said concept proved beneficial, then future development could be carried out on more popular platforms, namely the iPhone and similar products.

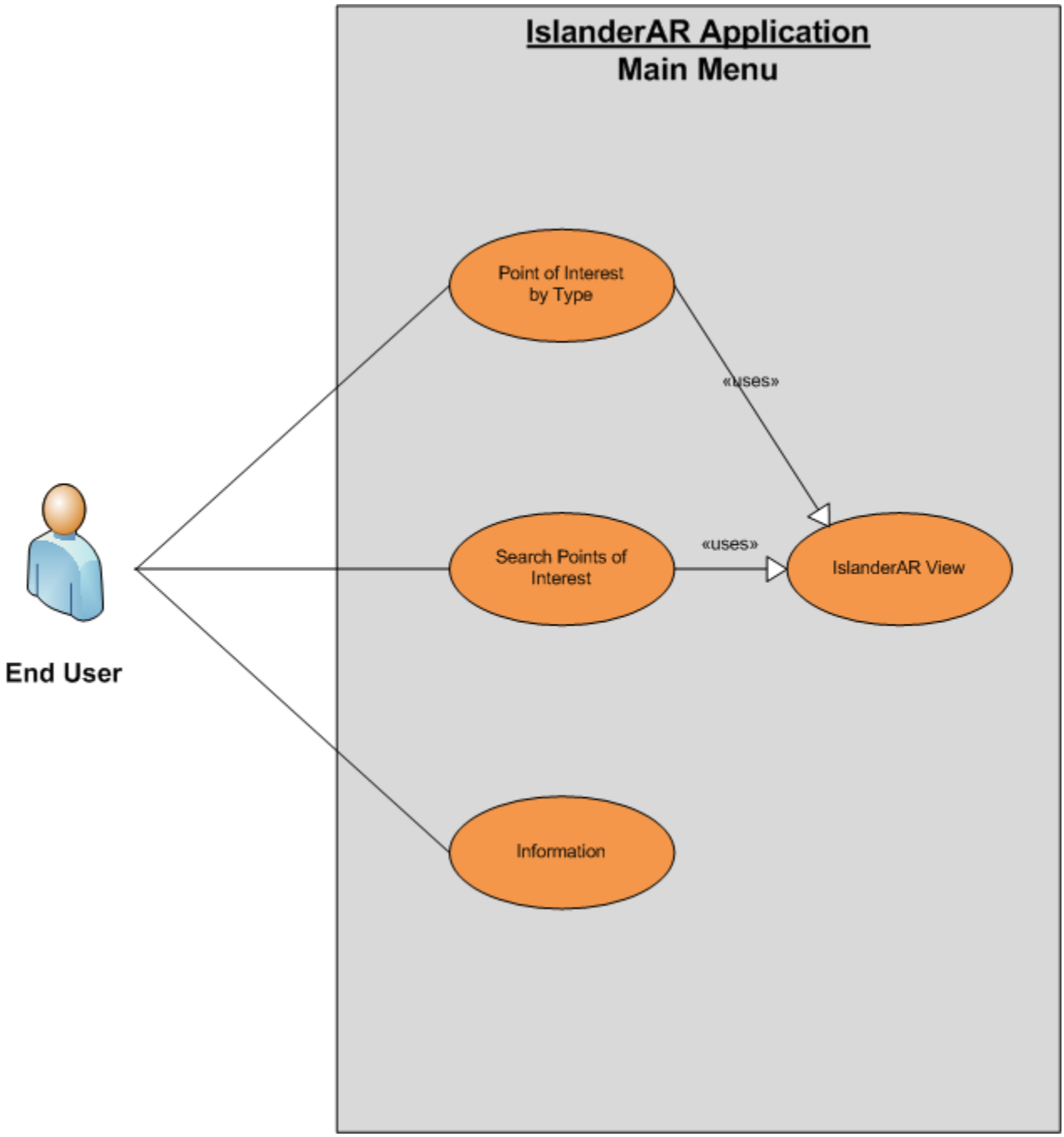
Ms. Bazan was very receptive to the proposal, and agreed to become the teams' client. They informed her of the next steps, contact from Dr. Mahdy, as well as requirement sessions, following a systems proposal.

Contact Information:

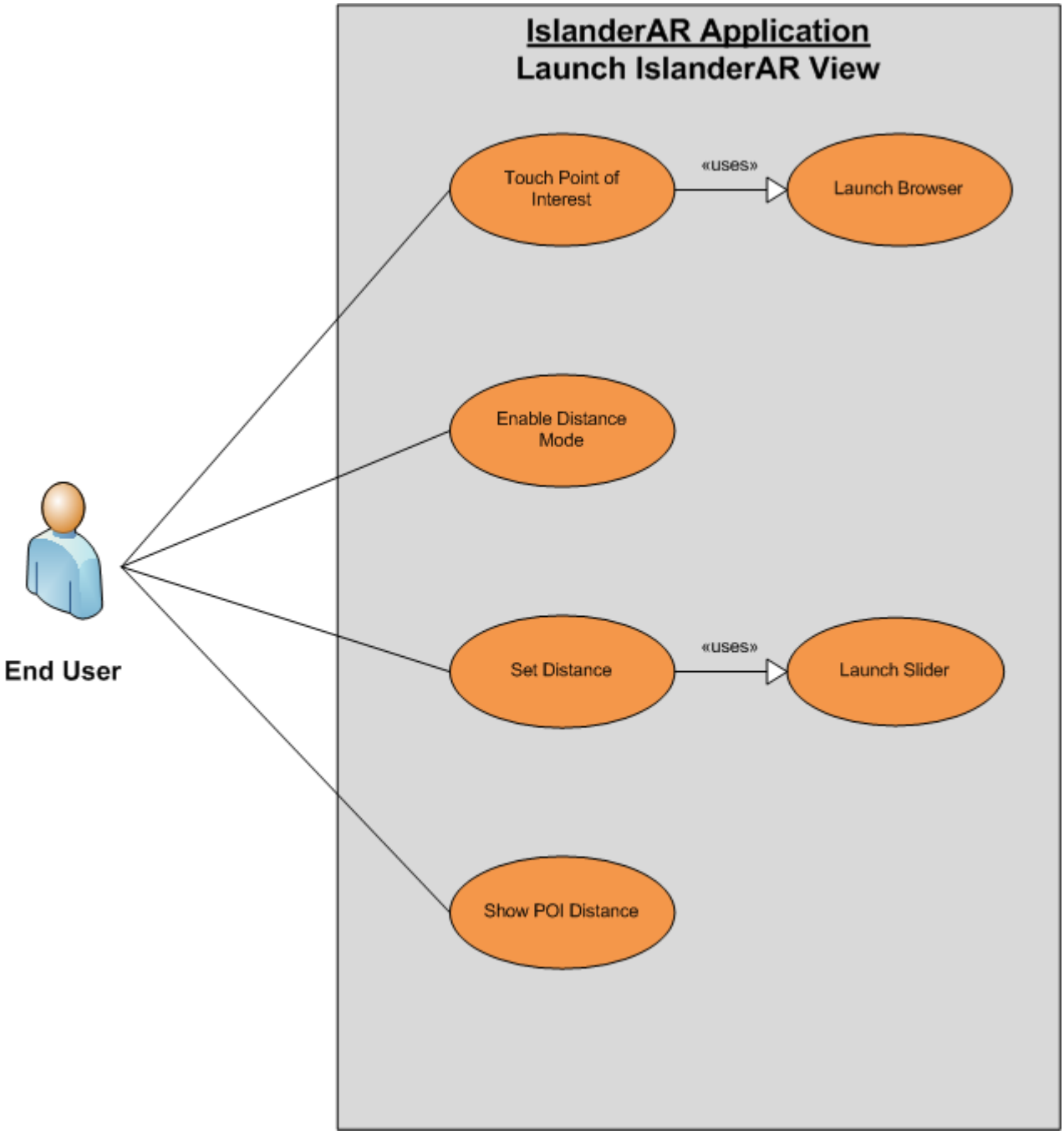
Leticia Bazan
Director – Office of Student Recruitment & New Student Programs
leticia.bazan@tamucc.edu
361.825.2292

Analysis

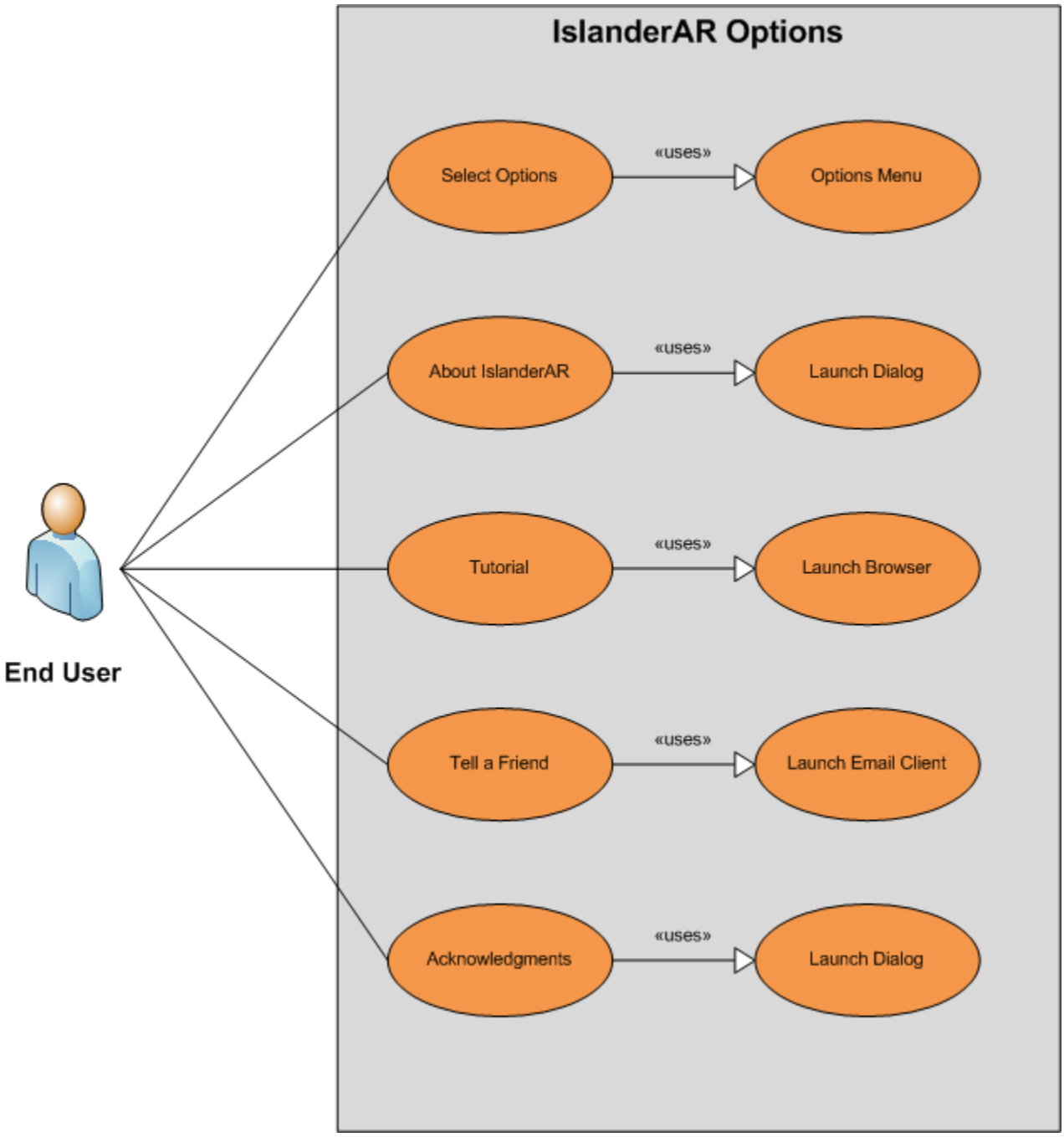
View Main Menu Use case



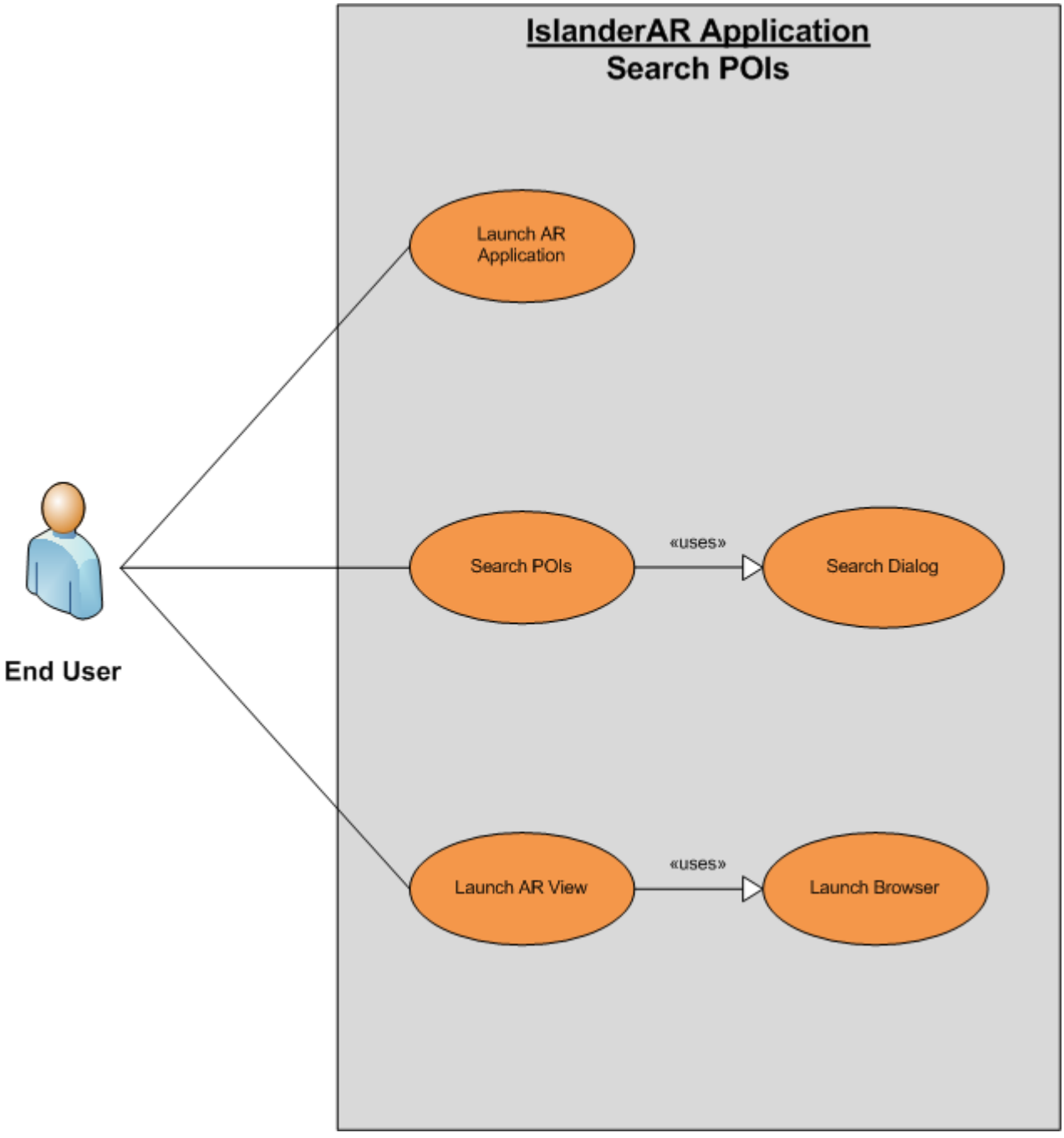
Launch IslanderAR View Use Case



View Options Use Case



Search Points of Interest Use Case



Use Case Realization

Point of Interest Use Case:

1. User A launches IslanderAR.
2. The **Splash** class displays the splash screen.
3. The **Splash** class launches the **Menu** class and ends.
4. User A selects “Points of Interest” from the menu.
5. The **Menu** class displays the Points of Interest by type selection menu.
6. User A selects “Campus Buildings”***** category and clicks OK.
7. The **Menu** class sends a query to the **QueryHandler** class.
8. The **QueryHandler** class creates an instance of the initialized database by passing the query to the DataBaseHandler class.
9. The **QueryHandler** populates the POI array list and passes it to the **AREngine** class.
10. The **AREngine** initiates the device listeners and passes the device listener input to the **PositionCalculation** class.
11. The **AREngine** class passes the POI instances to the **AugmentedView** class.
12. The **AugmentedView** class initiates the Draw method of the **POI** class.
13. The **POI** class initiates the DetermineLocation method of the **PositionCalculation** class.
14. The **POI** class initiates the DrawAtLocation method of the **GUITextBox** class to draw the visible POIs.
15. The **AugmentedView** class returns the drawn canvas to the **AREngine** class.
16. As User A moves the device, the **AREngine** class passes the new coordinates to the **AugmentedView** class, and steps 12 – 15 are repeated.
17. User A presses the device Menu button to launch the IslanderAR menu.
18. The **AREngine** is passed the event and calls the OnPrepareOptions method.
19. The **AREngine** calls the OnOptionsCreate method to display the IslanderAR menu.
20. User A selects “Set Distance” from the IslanderAR menu.
21. The **AREngine** creates a SeekBar from the **SeekBarListener**.
22. The **SeekBarListener** passes the listener to the **AREngine** isShowable method.
23. User A selects the desired distance and presses OK.
24. The **AREngine** updates the curDistPref variable.
25. The **AREngine** passes the new coordinate information to the **AugmentedView** class and steps 12 – 15 are repeated.
26. User A presses the device Menu button to launch the IslanderAR menu.
27. The **AREngine** is passed the event and calls the OnPrepareOptions method.
28. The **AREngine** calls the OnOptionsCreate method to display the IslanderAR menu.
29. User A selects “Show POI Distance” from the IslanderAR menu.
30. The **AREngine** class sets the ShowDistance variable to TRUE.
31. The **AREngine** class passes the new POI information to the **AugmentedView** class and steps 12 – 15 are repeated.
32. User A presses the device Menu button to launch the IslanderAR menu.
33. The **AREngine** is passed the event and calls the OnPrepareOptions method.
34. The **AREngine** calls the OnOptionsCreate method to display the IslanderAR menu.
35. User A selects “Enable Distance Mode” from the IslanderAR menu.
36. The **AREngine** class sets the DistanceMode variable to TRUE.
37. The **AREngine** class passes the new POI information to the **AugmentedView** class and steps 12 – 15 are repeated.
38. User A presses the device back button to end the IslanderAR application.

*The scenario remains the same for any or all of the POI type selections.

Search Point of Interest Use Case:

1. User A launches IslanderAR.
2. The **Splash** class displays the splash screen.
3. The **Splash** class launches the **Menu** class and ends.
4. User A selects "Search Point of Interest" from the menu.
5. The **Menu** class passes the user selection to the **AREngine** class.
6. The **AREngine** class calls the DIALOG_SEARCH method of the **Menu** class.
7. The DIALOG_SEARCH method calls the text Dialog method provided by the Operating System.
8. User A inputs their selection using the on-screen keyboard provided by the Operating System.
   ALTERNATIVE: User A inputs their selection using the device's physical keyboard.
9. The text Dialog method returns the input to the **Menu** class.
10. The **Menu** class sends a query to the **QueryHandler** class.
11. The **QueryHandler** class creates an instance of the initialized database by passing the query to the DataBaseHandler class.
12. The **QueryHandler** populates the POI array list and passes it to the **AREngine** class.
13. The **AREngine** initiates the device listeners and passes the device listener input to the **PositionCalculation** class.
14. The **AREngine** class passes the POI instance to the **AugmentedView** class.
15. The **AugmentedView** class initiates the Draw method of the **POI** class.
16. The **POI** class initiates the DetermineLocation method of the **PositionCalculation** class.
17. The **POI** class initiates the DrawLeftArrow or DrawRightArrow method of the **GUITextBox** class.
18. User A rotates in the direction indicated by the arrow until the POI falls within viewing range.
19. The **AugmentedView** class initiates the DrawAtLocation method of the **GUITextBox** class to draw the visible POI.

At this point the Use Case enters the Point of Interest Use Case at step 15.

Information Use Case:

1. User A launches IslanderAR.
2. The **Splash** class displays the splash screen.
3. The **Splash** class launches the **Menu** class and ends.
4. User A selects :Information" from the menu.
5. The **Menu** class displays the Information Menu.
6. User A selects "What is a Point of Interest"
7. The **AREngine** causes an Operating System dialog box to be displayed that defines a Point of Interest.
8. User A selects the OK button.
9. The dialog box closes and the Information menu receives focus.
10. User A selects "About" from the Information menu.
11. The **AREngine** causes an Operating System dialog box to be displayed that displays a brief message describing the IslanderAR application.
12. User A selects the OK button.
13. The dialog box closes and the Information menu receives focus.
14. User A selects "Tutorial" from the Information menu.
15. The **AREngine** launches the device web browser and passes the URL of the Tutorial website to it.
16. User A completes the tutorial and closes the web browser.
17. The Information menu receives focus.
18. User A selects "Tell a friend" from the Information menu.
19. The **AREngine** launches the device email client application.
20. User A composes an email and clicks "Send", or User A clicks "Cancel".
21. The email client application closes and the Information menu receives focus.
22. User A selects "Acknowledgements" from the Information menu.
23. The **AREngine** launches the device web browser and passes the URL of the Acknowledgement website to it.
24. User A views the web page, and closes the web browser.
25. The Information menu receives focus.
26. User A selects Done and returns to the IslanderAR menu.

Design Rationale and Decisions

Initial Design Considerations:

The Android operating system was a brand new development platform to every member of the team. Therefore, the learning curve associated with the new environment was quite substantial. This is evident in the fact that the team's initial development plans included utilizing the Wikitude World Browser augmented reality application, and later, the Gamaray open source augmented reality application.

The team's unfamiliarity with the operating system and associated code lead them into the direction of simply implementing found code. However, it was decided late in the semester that this was an unacceptable solution. At that point, the development of a custom augmented reality application began. The experience gained from working with Android source code examples, in particular the Gamaray open sourced application, paved the way for the custom application to emerge as successfully as it did. Therefore, the initial design and implementation considerations in the earlier part of the semester were a necessary evolutionary step taken towards a successful completion of the actual project.

After the choice was made to develop a custom application for the Android OS, the development team set to work designing and implementing the classes and structured storage needed the realize the project's use cases.

The first step to this process was to identify the actual Android framework specific classes that would need to extended or implemented in order to integrate our design into the operating system as seamlessly as possible. It was our intention to create an application that had the "look & feel" of an Android application, while being specific to the Texas A&M-Corpus Christi campus.

In order to fully understand how we should implement the Android framework, we had to research and test the application programming interface available to us. This began by understanding the architecture of the Android operating system, as seen in **Figure D-1** the below.

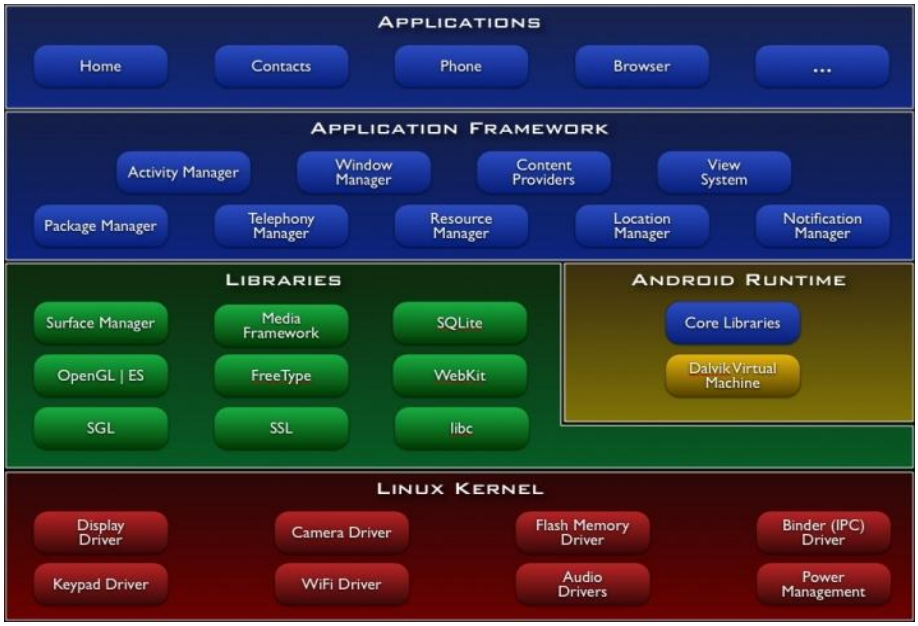


Figure D-1

Upon realizing the framework available to us, we then needed to understand the lifecycle of the typical android process, called an activity. An application is typically made up of several activities all grouped together into a task of activities. A single activity is usually represented by a single user interface screen that “does work” or provides for user input and output. In order to successfully implement our application, we needed to decide which classes we developed should be activities.

Figure D-2 presents a visual diagram of the methods that are available to the developer during multiple phases of the Android activity lifecycle. These methods are typically called implicitly when an activity moves from one state to the next, as seen in **Figure D-3**.

In order to successfully implement and extended Android activity, the polymorphic methods listed should be overridden with custom development code that describes what actions should be taken in the main thread of execution within the current activity.

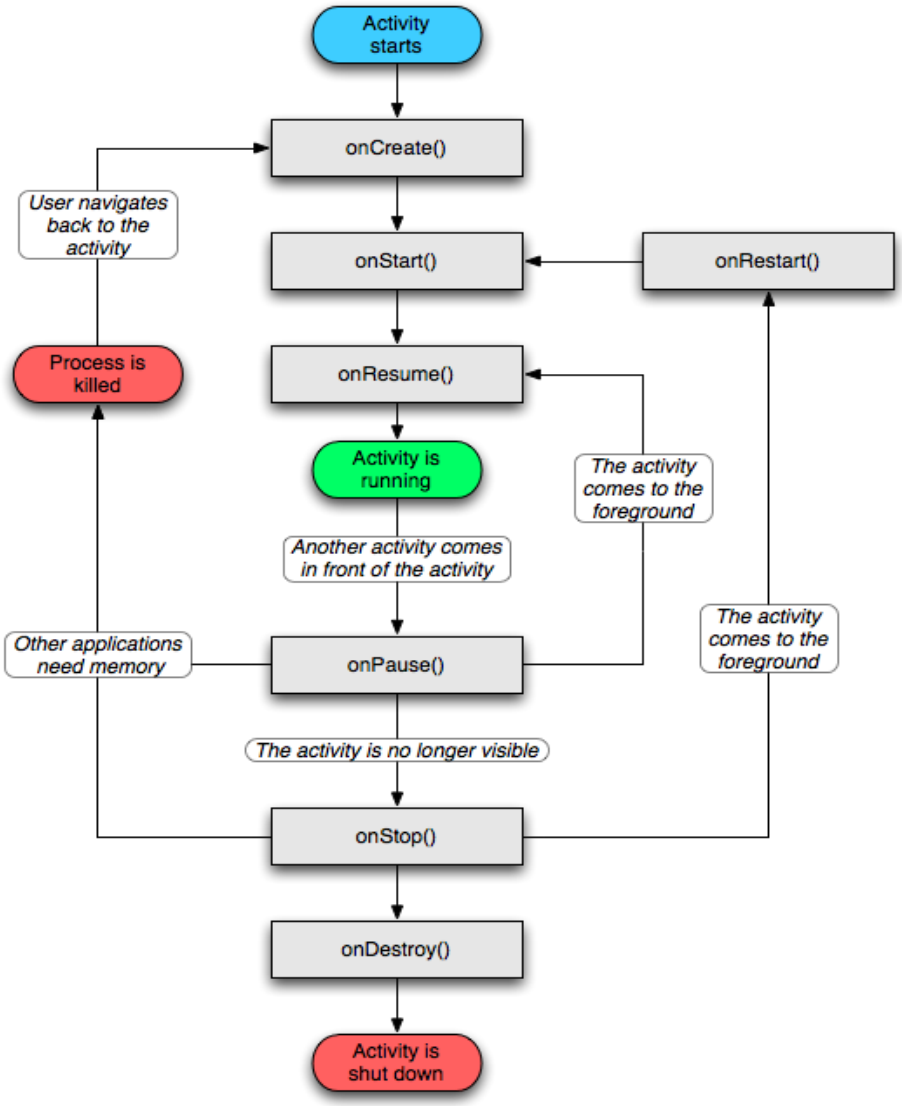


Figure D-2

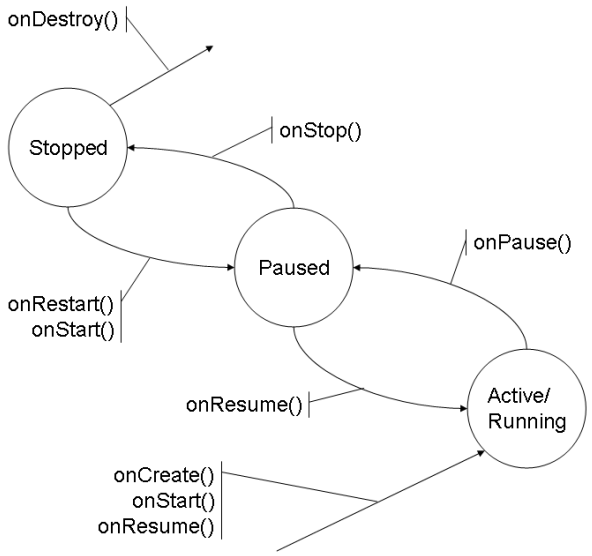


Figure D-3

Design Rational & Decisions:

With the aforementioned platform specific requirements in mind, we designed a total of fourteen classes that extended and implemented a total of nine different Android framework objects.

Additionally, the primary decisions of how the application should work were also designed at this point. Those decisions mainly effected how the program was to operate in terms of determining the device's current location, and the location of every point of interest that the user requested. The steps involved with accomplishing said task are as follows.

1. The device's GPS would be registered for updates with the operating system.
2. Additionally, the device's sensors were also registered with the operating system.
3. Once a GPS signal was available, the distance to every user selected point of interest would be calculated.
4. Distance calculations were completed by using the law of haversines trigonometric function, and two pairs of POI coordinates in decimal degree format.
5. The device's degrees were determined by the GPS.
6. The specific POI degrees were read from the database, and stored within the instance of the POI entity.
7. The calculated distance would also be stored as an attribute of the POI entity.
8. The bearing to the POI of interest would then be calculated by determining the difference between the device's current bearing, and the point of interest. Again, implementing some simple trigonometry.
9. The bearing to the POI would then be stored in the specific POI instance.

With this information, the application could determine a screen position for the point of interest based upon a few more calculations. The screen position calculation is also considered to be a major design decision that was completed at the beginning of the workflow.

The screen position calculation takes several factors into account. The first factor was the known screen width of 480 pixels, when the device is held in a horizontal position. The second factor was the focal length of the device's camera. After researching the specific focal length for some time without an appropriate answer, the field of view was assumed to be around 50 degrees. This assumption was based upon a field test of the device, where the team took measurements of number of degrees it took for an object to appear and disappear off of the screen. The last factor was the bearing to the specific point of interest, a value we derived at run time. Therefore, with these values in mind, we can calculate a screen position for each point of interest based upon the exact degree to said point, and a positive and negative 25 degree buffer to simulate the total ~50 degree field of view. This is accomplished by dividing the 480 pixel screen into 50 equal parts along the x axis, and displaying the POI GUITextBox object accordingly.

Class Enumeration and Specific Design Details:

1. AREngine **extends** Activity

   The main activity of the application. This class provides the base for which most of the other classes are built and instantiated.

2. AugmentedView **extends** View

   This is the view that is merged with the CameraView to create the augmented reality aspect of the application.

3. CameraView **extends** SurfaceView **implements** SurfaceHolder.Callback

   The CameraView class features all of the framework elements necessary to capture the camera stream and apply it to the extended surface view.

4. CompassListener **implements** SensorListener

   The CompassListener implements the SensorListener object in order to retrieve the array of values that represent the degree angles that the device is currently positioned in. According to the Android development guide, the SensorListener class is a deprecated version from an earlier API that has been replaced by the SensorEventListener class. Due to errors concerning the SensorEventListener class, it was decided to simply use the older SensorListener class, which was free of erroneous data. Finally, the CompassListener class also makes the call to the AugmentedView to postInvalidate, which requires that the view re-draw itself. This is done every time that the CompassListener class receives new input from the sensors.

5. DataBaseHandler **extends** SQLiteOpenHelper

   The DataBaseHandler class is tasked with opening and returning an instance of the SQLite database that is used in the application. This class also allows the application to ship with an external SQLite database that is then built locally the first time the application is run. This local database is necessary because Android applications require that any associated database needs to be stored in a task specific location on the disk.

6. GPSListener **implements** LocationListener

   The GPSListener class implements all of the required methods to successfully read an Android Location object from a GPS update. For this application, the team has also implemented location change checking, to reduce the number of updates when the device is not moving. This feature allows the application to limit the resource intensive distance calculations to when the device has physically moved a noticeable distance. The end result of these restrictions is reduced processor time spent on floating point calculations of degree decimal locations.

7. GUITextBox **extends** Drawable

   Extending an Android Drawable object allows the GUITextBox to be treated as an extension of a view. This allows the application to pass a canvas to the object, and have it drawn upon the given view canvas. Additionally, the GUITextBox class builds itself from standard Android Drawable objects, such as rounded rectangle and paints. The class implements some arithmetic procedures to determine the appropriate box size based upon the text attributed to the message attribute of the class. When the GUITextBox draws itself, it does so around a set of x, y screen coordinates that are supplied to the setPosition method. The passed x, y coordinate values become the location of the middle of the textbox. This feature allows the application to determine what POI GUITextBox has been touched based upon a radius around the exact center of the text box.

8. Horizontal_Browser **extends** Activity

   This class was implemented as an activity because the standard Android Browser application does not have a feature to launch in a horizontal configuration. One of the requisites of the application is that our custom web pages launch in a horizontal screen configuration so that the user does not need to constantly rotate the device. In order to implement this feature, we simply created our own browser using the available framework.

9. Menu **extends** Activity

   The Menu activity is the second activity that the user will come in contact with after the application launches. The menu provides several Android dialog boxes that allow for multiple forms of user input. All input supplied to this class is forwarded to the QueryHandler class for query building and database queries.

10. PointOfInterest

    The PointOfInterest class is the main entity object in the application. All of the work done is in order to simply display these objects, and their respective boundary objects to the screen.

11. PositionCalculation

    The PositionCalculation object is a control class of static methods that acts as a library of procedures and functions used by the other objects contained within this application. All non-trivial calculations performed by this program are housed within this class.

12. QueryHandler

    Similar to the PositionCalculation class, the QueryHandler object contains several static method that act as a library of procedures and functions for the other instantiated objects to utilize. Although QueryHandler does not perform any calculations, its methods are functionally cohesive and provide the other objects in the application the functions needed to create and query SQL statements against the SQLite database.

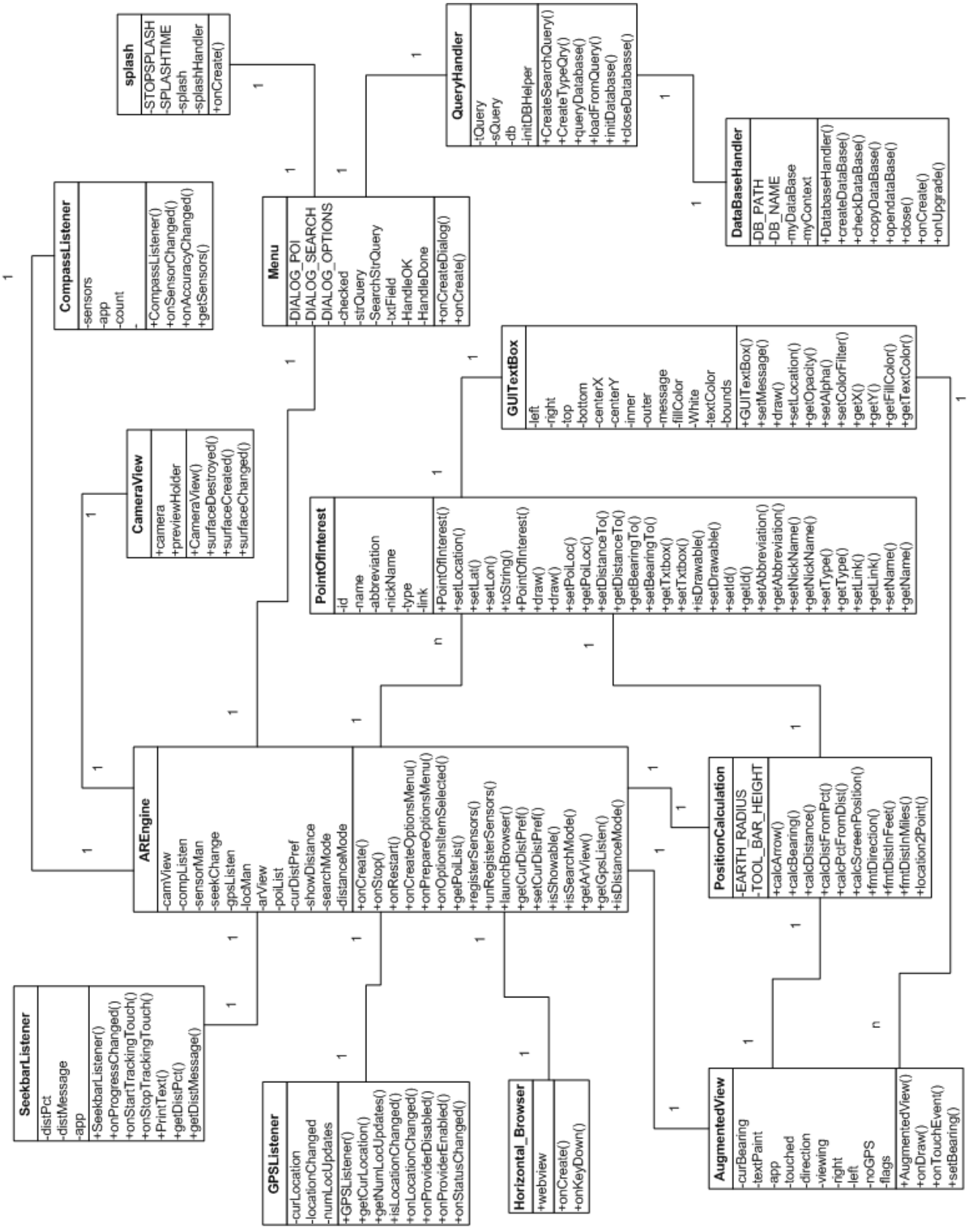
13. SeekbarListener **implements** SeekBar.OnSeekBarChangeListener

    This class is used to allow for the integration of an Android SeekBar object within the source code.

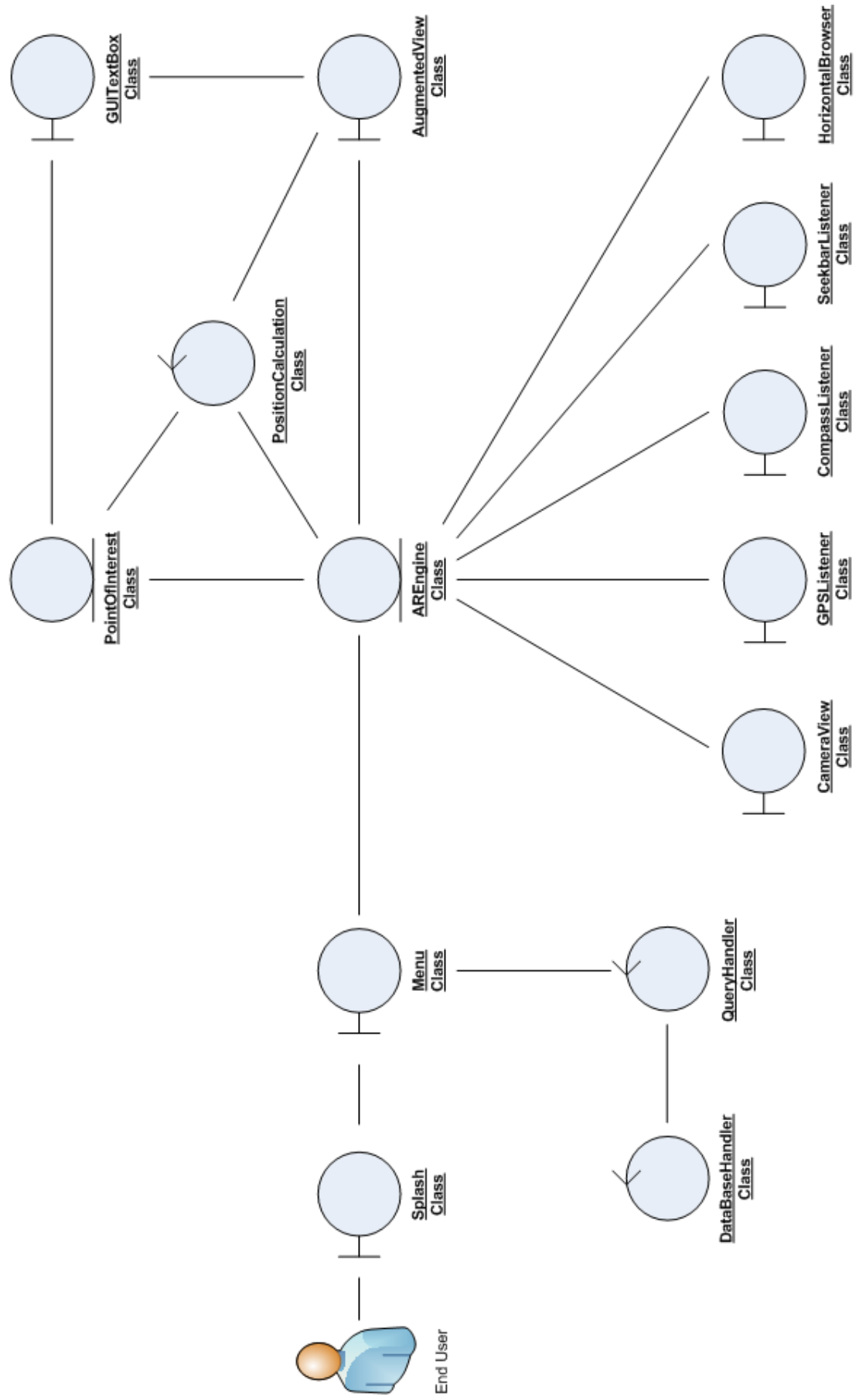
14. splash **extends** Activity

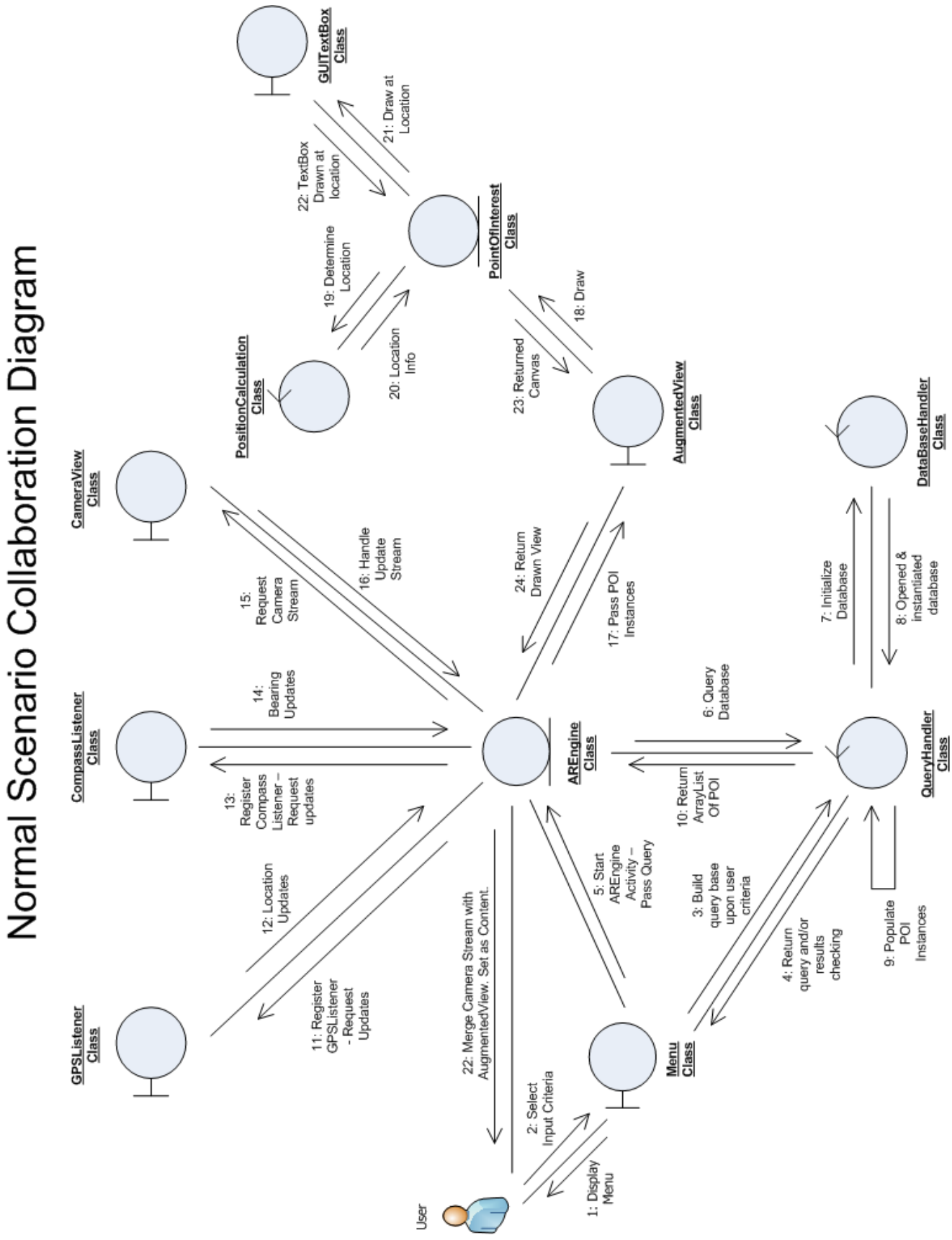
    The splash activity class exists to give the user a three second splash screen at the launch of the application.

Design Diagrams

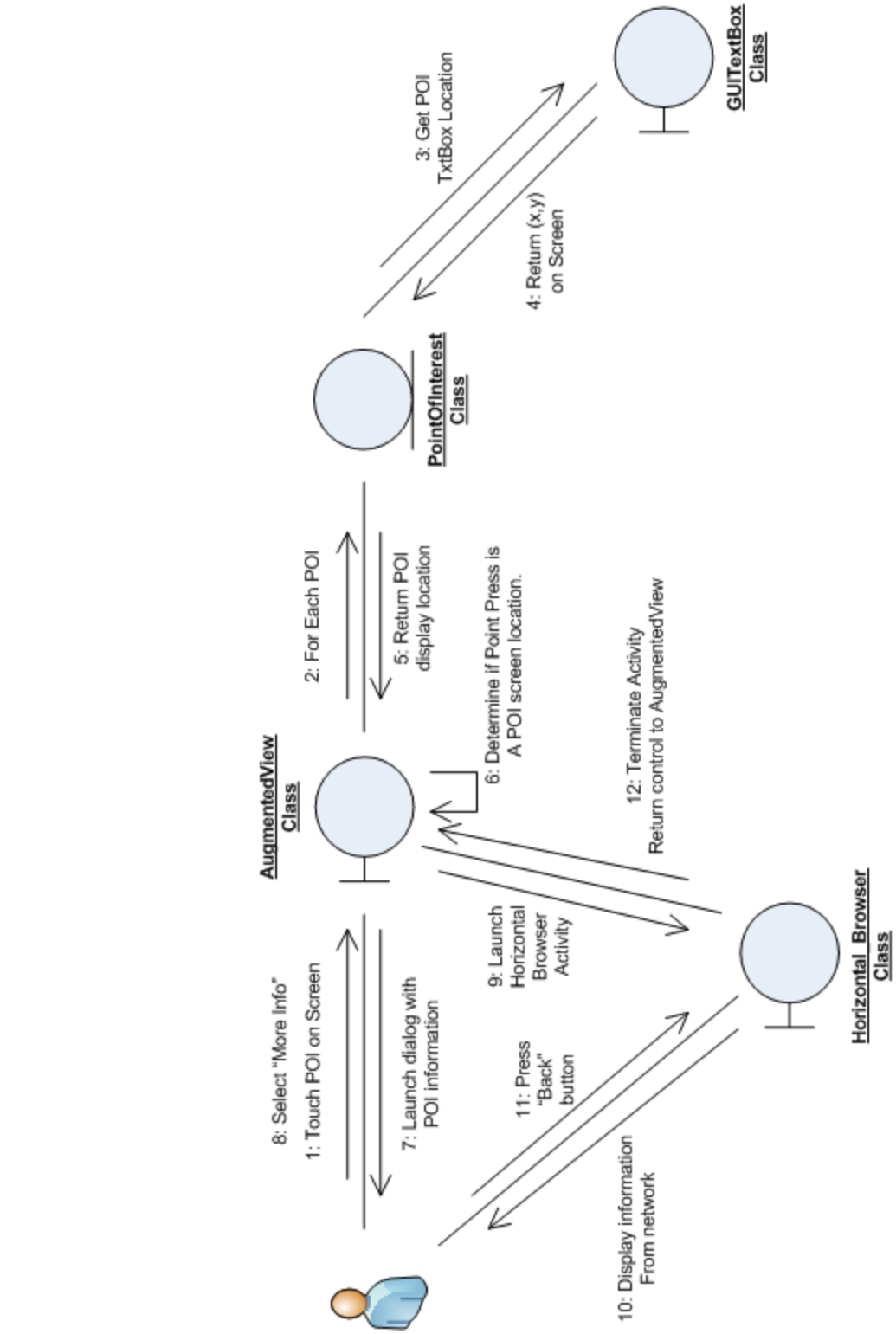

splash
-STOPSPLASH
-SPLASHTIME
-splash
-splashHandler
+onCreate()
QueryHandler
-tQuery
-sQuery
-db
-initDBHelper
+CreateSearchQuery()
+CreateTypeQry()
+queryDatabase()
+loadFromQuery()
+initDatabase()
+closeDatabase()
DataBaseHandler
-DB_PATH
-DB_NAME
-myDataBase
-myContext
+DatabaseHandler()
+createDataBase()
+checkDataBase()
+copyDataBase()
+opendataBase()
+close()
+onCreate()
+onUpgrade()
CompassListener
-sensors
-app
-count
-
+CompassListener()
+onSensorChanged()
+onAccuracyChanged()
+getSensors()
Menu
-DIALOG_POI
-DIALOG_SEARCH
-DIALOG_OPTIONS
-checked
-strQuery
-SearchStrQuery
-txtField
-HandleOK
-HandleDone
+onCreateDialog()
+onCreate()
CameraView
+camera
+previewHolder
+CameraView()
+surfaceDestroyed()
+surfaceCreated()
+surfaceChanged()
GUITextBox
-left
-right
-top
-bottom
-centerX
-centerY
-inner
-outer
-message
-fillColor
-White
-textColor
-bounds
+GUITextBox()
+setMessage()
+draw()
+setLocation()
+getOpacity()
+setAlpha()
+setColorFilter()
+getX()
+getY()
+getFillColor()
+getTextColor()
PointOfInterest
-id
-name
-abbreviation
-nickName
-type
-link
+PointOfInterest()
+setLocation()
+setLat()
+setLon()
+toString()
+PointOfInterest()
+draw()
+draw()
+setPoiLoc()
+getPoiLoc()
+setDistanceTo()
+getDistanceTo()
+getBearingTo()
+setBearingTo()
+getTxtbox()
+setTxtbox()
+isDrawable()
+setDrawable()
+setId()
+getId()
+setAbbreviation()
+getAbbreviation()
+setNickName()
+getNickName()
+setType()
+getType()
+setLink()
+getLink()
+setName()
+getName()
AREngine
-camView
-compListen
-sensorMan
-seekChange
-gpsListen
-locMan
-arView
-poiList
-curDistPref
-showDistance
-searchMode
-distanceMode
+onCreate()
+onStop()
+onRestart()
+onCreateOptionsMenu()
+onPrepareOptionsMenu()
+onOptionsItemSelected()
+getPoiList()
+registerSensors()
+unRegisterSensors()
+launchBrowser()
+getCurDistPref()
+setCurDistPref()
+isShowable()
+isSearchMode()
+getArView()
+getGpsListen()
+isDistanceMode()
PositionCalculation
-EARTH_RADIUS
-TOOL_BAR_HEIGHT
+calcArrow()
+calcBearing()
+calcDistance()
+calcDistFromPct()
+calcPctFromDist()
+calcScreenPosition()
+fmtDirection()
+fmtDistInFeet()
+fmtDistInMiles()
+location2Point()
SeekbarListener
-distPct
-distMessage
-app
+SeekbarListener()
+onProgressChanged()
+onStartTrackingTouch()
+onStopTrackingTouch()
+PrintText()
+getDistPct()
+getDistMessage()
GPSListener
-curLocation
-locationChanged
-numLocUpdates
+GPSListener()
+getCurLocation()
+getNumLocUpdates()
+isLocationChanged()
+onLocationChanged()
+onProviderDisabled()
+onProviderEnabled()
+onStatusChanged()
Horizontal_Browser
+webview
+onCreate()
+onKeyDown()
AugmentedView
-curBearing
-textPaint
-app
-touched
-direction
-viewing
-right
-left
-noGPS
-flags
+AugmentedView()
+onDraw()
+onTouchEvent()
+setBearing()

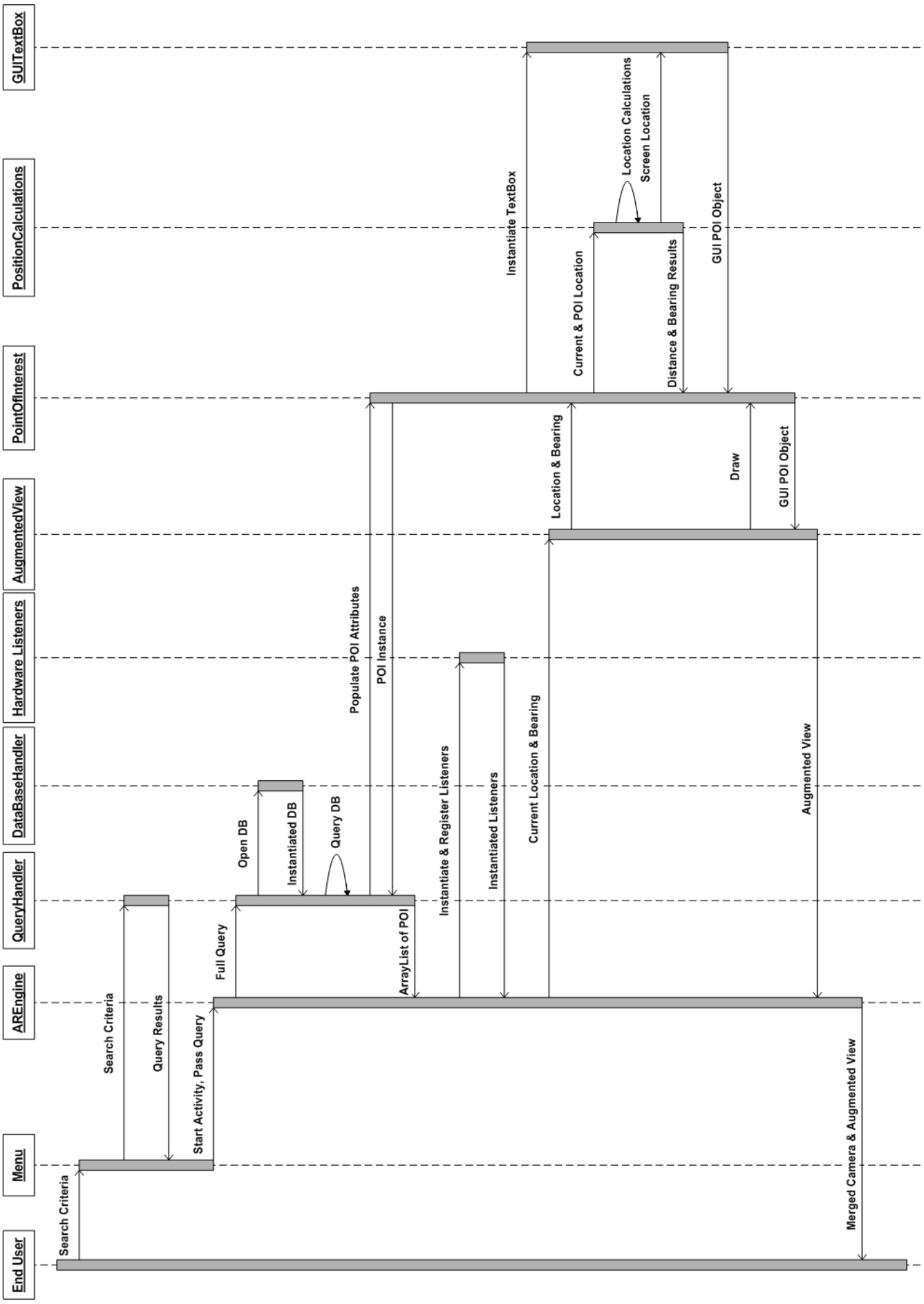
Class Diagram




Normal Scenario Collaboration Diagram
GPSListener Class
CompassListener Class
CameraView Class
PositionCalculation Class
GUITextBox Class
PointOfInterest Class
AugmentedView Class
AREngine Class
Menu Class
QueryHandler Class
DataBaseHandler Class
User
1: Display Menu
2: Select Input Criteria
3: Build query base upon user criteria
4: Return query and/or results checking
5: Start AREngine Activity – Pass Query
6: Query Database
7: Initialize Database
8: Opened & instantiated database
9: Populate POI Instances
10: Return ArrayList Of POI
11: Register GPSListener - Request Updates
12: Location Updates
13: Register Compass Listener – Request updates
14: Bearing Updates
15: Request Camera Stream
16: Handle Update Stream
17: Pass POI Instances
18: Draw
19: Determine Location
20: Location Info
21: Draw at Location
22: TextBox Drawn at location
23: Returned Canvas
24: Return Drawn View
22: Merge Camera Stream with AugmentedView. Set as Content.

Exceptional Scenario:
View Point of Interest Information with Web Information


AugmentedView Class
PointOfInterest Class
GUITextBox Class
Horizontal Browser Class
2: For Each POI
5: Return POI display location
3: Get POI TxtBox Location
4: Return (x,y) on Screen
6: Determine if Point Press is A POI screen location.
12: Terminate Activity Return control to AugmentedView
8: Select "More Info"
1: Touch POI on Screen
7: Launch dialog with POI information
9: Launch Horizontal Browser Activity
11: Press "Back" button
10: Display information From network


End User
Menu
AREngine
QueryHandler
DataBaseHandler
Hardware Listeners
AugmentedView
PointOfInterest
PositionCalculations
GUITextBox
Search Criteria
Search Criteria
Query Results
Start Activity, Pass Query
Full Query
Open DB
Instantiated DB
Query DB
ArrayList of POI
Populate POI Attributes
POI Instance
Instantiate & Register Listeners
Instantiated Listeners
Current Location & Bearing
Augmented View
Location & Bearing
Draw
GUI POI Object
Instantiate TextBox
Current & POI Location
Location Calculations
Screen Location
Distance & Bearing Results
GUI POI Object
Merged Camera & Augmented View

Database

Database Design Rationale

Due to the nature of the information contained in the database, and how that information is used by the end-user has played large part in the final design of the Islander AR relational database.

The informational needs of the end-user are quite simple, and just require the ability to access the database for read only purposes. The database is currently not transactional. Since, the end-user is not allowed to perform database transactions; database quality will not be affected by the end-user through the Islander AR application.

The information contained within the database is type specific geo-located points of interest located on the Texas A&M-Corpus Christi campus, which inherently should not change at all. The required attributes for each record can be found in the data dictionary **Figure D-5**.

The database has been normalized into the third normal form (3NF) stage, which indicates the removal of all redundant data, as well as any functional, and transitive dependencies from the database. Having the database in third normal form (3NF) does require the use of several relational join operations to produce specific results, which will cause more resources to be used in order to respond to end-user queries. However, the database has been designed to perform relatively well considering the limited resources in which we have to work with on the device. The database has been kept simple with the intent of maximizing device performance. This can be seen in **Figure D-4** in the crow's foot entity-relational model.

Another consideration to take into account includes the use of SQLite. The SQLite software library has already been pushed onto the device and is used in many of the standard functions currently found on the Google G1 phone today (i.e. contact information). SQLite is a self-contained, serverless, zero-configuration, transactional SQL database engine. There are limitations associated with the use of SQLite; it is of course a scaled down implementation of SQL92, and some features are not permitted (i.e. right outer join, full outer join). A full explanation of all features not supported by SQLite can be found at: http://www.sqlite.org.

Crow's Foot Entity-Relational Model

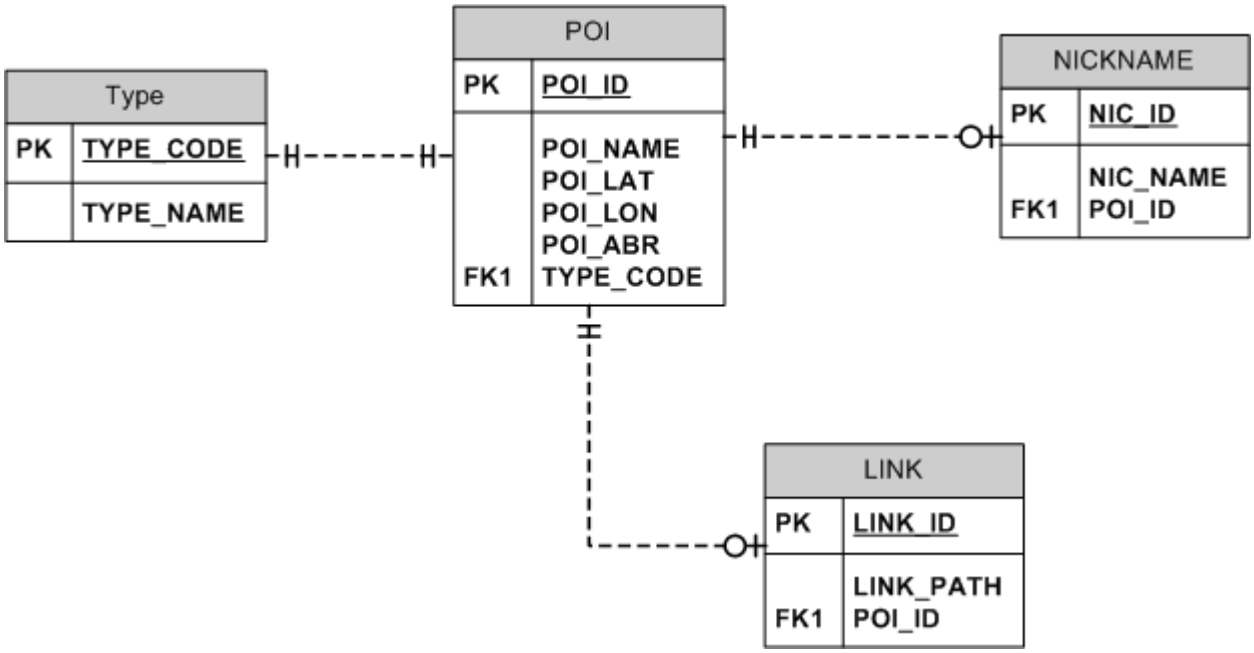


Figure D-4

Data Dictionary

TABLE NAME	ATTRIBUTE NAME	DESCRIPTION	DATATYPE	FORMAT	RANGE	REQUIRED	PK OR FK	FK REFERENCED TABLE
TYPE	TYPE_CODE	Point of interest type	INTEGER	999	1-999	Y	PK	
	TYPE_NAME	Point of interest name	TEXT	XXXXXXXX		Y		
POI	POI_ID	Point of interest id	INTEGER	999	1-999	Y	PK	
	POI_NAME	Point of interest name	TEXT	XXXXXXXX		Y		
	POI_ABR	Point of interest abbreviation	TEXT	XXXXXXXX		Y		
	POI_LAT	Point of interest latitude	REAL	99.999999	-90 - 90	Y		
	POI_LON	Point of interest longitude	REAL	-99.999999	-180 - 180	Y		
	TYPE_CODE	Point of interest type	INTEGER	999	1-999	Y	FK	TYPE_CODE
LINK	LINK_PATH	Point of interest http link	TEXT	XXXXXXXX		Y		
	LINK_ID	Link id	INTEGER	999	1-999	Y	PK	
	POI_ID	Point of interest id	INTEGER	999	1-999	Y	FK	POI_ID
NICKNAME	NIC_NAME	Point of interest nickname	TEXT	XXXXXXXX		Y		
	NIC_ID	Nickname id	INTEGER	999	1-999	Y	PK	
	POI_ID	Point of interest id	INTEGER	999	1-999	Y	FK	POI_ID

Figure D-5

Implementation

Implementation:

The development of the IslanderAR application required the use of several resources. The IslanderAR application has been implemented using the Android software stack for mobile devices that includes the Android operating system which is a scaled down version of Linux, middleware and other key applications. The Android SDK, along with the Eclipse integrated development environment were key components to developing IslanderAR application on the Android platform using the Java programming language.

Features of the Android software stack:

- **Application framework** enables reuse and replacement of components
- **Dalvik virtual machine** optimized for mobile devices
- **Integrated browser** based on the open source WebKit engine
- **SQLite** for structured data storage
- **Rich development environment** this includes a device emulator, debugging tools

XML (Extensible markup Language) was used to present essential information about the application to the Android system before it can run any of the application's code.

Important uses of XML:

- Names the Java package for the application
- Describes components of the application – activities, services of the application
- Determines which processes will host application components
- Declares permissions
- Lists instrumentation classes that provide profiling and other information as the application is running
- Declares the minimum level of the Android API that the application requires
- List libraries that the application must be linked against

SQLite was another major component used in the creation of the IslanderAR application. The Android API contained support for creating and using SQLite databases. The SQLite database system allows for the storage of complex collections of data. Also, used was the SQLite database Browser GUI to design and create the IslanderAR database.

A revision control system (svn server) was implemented to maintain current and historical versions of source code. The team built a dedicated server running the latest stable version of Debian Linux 5.0 as the base operating system for the aforementioned svn server.

Omondo UML 2.2 eclipse plug-in trial version was also used to create a class diagram.

Testing Activities

Testing Activities:

The recommended development environment for any Android application development is the Eclipse Java IDE. By integrating the Android Development Tools (ADT) plugin, the user has access to extensions for the IDE that allows easy application development and debugging.

The plugin provides access to the Android Development Tools which consist of the ability to view application thread and process information directly from Eclipse. The plugin also provides the ability to take screenshots of the device screen. It also provides an Android Development Device emulator that allows the developer to run applications for testing and debugging purposes.

All testing using the Wikitude API took place using the Android™ emulator whenever possible. Once the initial application was developed sufficiently, all testing took place on the device. The same is true of the switch to Gamaray.

Once the team began developing our own code, we continued to use both methods. We also implemented user testing by allowing fellow students use the device and application. We also showed the application to curious on-lookers. In both cases we received useful feedback. The development process and the use of campus facilities allowed us to implement several ideas almost immediately in a rudimentary form. We were then able to test the code and refine it as needed.

Other than the known issues with interference and the occasional loss of sensor calibration, the application performs at or above expectations and appears to meet all of the project requirements.

Team AR Meeting Minutes

Group Activities

Activity	Time	Total Time
Scheduled Sunday meetings	~ 7 hours	94 hours
Scheduled presentation meetings	~ 4 hours	28 hours
Presentation rehearsals	~ 2 hours	12 hours
Ad-hoc meetings (after class, morning meetings, as needed)	~ 1 hour	15 hours
Total group project time:		149 hours

Team Meeting, 11 September, 2009

Attendees:
Ryan Egger
Patrick Krepps
Marco Longoria
Patrick O'Neal

The team met to discuss ideas for a capstone project.

The first item was the decision to try to develop a mobile application using the Android development system. Factors regarding this decision were that the system is open source and compatible with many platforms. Alternatives that were considered included building an application on the Layar development engine, and applications for the iPhone. The Layar system was not entirely ruled out. The iPhone was decided against because it limited our choices to a single platform.

We then discussed several application ideas. Among these were a GPS based information system that could provide information to the user such as current traffic conditions, tours of museums and/or parks, and augmented reality systems. Other ideas discussed were bluetooth information systems, such as something that could alert the user to parking conditions as they approached campus. There were several other ideas thrown out that have not been captured in these minutes.

In the end, we chose to take our ideas to Dr. Mahdy for advice on feasible projects that could be completed during the semester. Patrick Krepps was tasked with contacting Dr. Mahdy and setting up the meeting. Dr. Mahdy replied back indicating availability on Monday, 14 September, 2009 at 1100.

Team Meeting, 14 September, 2009

Attendees:
Ryan Egger
Patrick Krepps
Patrick O'Neal

Advisor:
Dr. Ahmed Mahdy

The team met with Dr. Mahdy to discuss several proposals, and to get his advice on the feasibility of our ideas. In the end we settled on trying to develop an information type of system based on bluetooth technology to provide information to user platforms independent of access to either a cellular or wi-fi signal. We also agreed to have a concrete proposal ready by class time on Thursday, 17 September, 2009.

After our discussions with Dr. Mahdy, we reconvened to further refine our ideas. We would like to try to develop a system that could be used in any setting, in which bluetooth transmitters would be able to provide information to the platform, such as in an art museum. As the user traversed the museum, each display would be tied to a transmitter in the area that would provide the user with a detailed description of the work. The system could also provide internet sites for the user to access to obtain additional information. This information could be provided in such a way that the user could take advantage of it when they then had access to the internet.
We also discussed implementing a similar system in a concert hall setting. As the user is enjoying the concert, our system could be providing information about the piece, such as name, artist, lyrics, etc. The user could also be presented with internet sites where they could purchase the music they are listening to.

At this point, the team is tasked with trying to refine our ideas into a solid idea so we can present it to Dr. Mahdy. We need to meet at least one more time before then.

Team Meeting, 20 September, 2009

Attendees:
Ryan Egger
Patrick Krepps
Patrick O'Neal

We began by discussing how we would like to implement our idea, and potential clients. We agreed that we should probably start with a rapid prototype of our envisioned system. There were several reasons for this decision. We need to show that we can access the GPS system as we need to. We need to show that we can access the mobile device's compass, and we need to show that we can overlay points of interest over the camera image on the device screen. We realize we are probably going to deliver a proof of concept, rather than a full-fledged, shelf-ready application.

We discussed application requirements and expectations, but postponed this discussion until we have a client. We will also have to start building the database of POI as soon as client determination has been made. We will also have to develop business requirements quickly.

We discussed potential clients. Our primary focus is on the campus New Student Programs (http://tour.tamucc.edu/). We also discussed alternative clients. These include: CCISD and locations/tours of campuses; the USS Lexington; the Corpus Christi RTA with bus stop locations/schedules; the Corpus Christi parks department; the Corpus Christi Chamber of Commerce with tourist Points of Interest; and local realtors.

We also discussed developing a time line of project milestones, and/or a rough schedule. We discussed meeting very soon with our laptops so that we can ensure we all have the same development kit as well as other software. We will most likely develop our application using the Wikitude API, developed on the Android platform. We will also try to keep track of all resources we use, and the hours we put into the project.

Project specific notes:
We want to be able to filter POI by type, content, and range (distance from user to POI).
We want to be able to access GPS, compass, and camera systems.

Action Items:
Patrick O to create google group to use for inter-meeting communication - Done
Patrick K to email Dr. Mahdy with potential clients.
All to research what we envision to show a proof of concept of our proof of concept application
All to post schedules to google group
All to choose a team name - Done, we are the AR Team or ART

Team Meeting, 04Oct, 2009

Attendees:
Ryan Egger
Patrick Krepps
Marco Longoria
Patrick O'Neal

Patrick O'Neal and Ryan Egger have made significant progress on application development. They have managed to produce a working prototype of the system that takes a GPS reading, and then displays one of four Points of Interest in the database. It also appears that the ability for the user to link to a POIs webpage is quite possible. The prototype suffers from one major problem – the GPS reading is taken when the application initially launches, and is not updated. We have not been able to find a way to update the reading, but we have not ruled out the possibility of either automatic updates, or manual updates in which the user selects an update button.

UPDATE: Upon mention the above problem to Dr. Mahdy, he made us aware of a service known as SkyHook Wireless. This service offers position triangulation by coordinating both cellular tower signals and known Wi-Fi access points. It appears this service may provide a solution to our positioning problem.

The team also developed a set of structured requirements. Due to the lack of contact from our chosen client, we have had to generate requirements from our point of view, and including the input we received from Laura Villalobos.

We also assigned primary tasks:
Documentation - Patrick K
Code Development - Patrick O and Ryan
Design – Marco Longoria.

The team also chose to continue to develop the application using the Wikitude API.

Team Meeting, 08 Nov, 2009

Attendees:
Ryan Egger
Marco Longoria
Patrick Krepps
Patrick O'Neal

The overwhelming amount of code involved in running Gamaray has become too much of a hindrance. It introduces a massive amount of unused code that must be loaded onto the device. As a result, earlier this week the team decided to drop the Gamaray development system in favor of completely custom code. Patrick O'Neal has already determined that it is possible to receive the necessary location, and other position/orientation information to make the system work.

Ryan has done a large amount of work on implementing the database into the application, and Marco has been doing the data entry. Patrick K has continued to develop documentation. At this point code development is outpacing that activity but the basic design remains the same. It is just a matter of making several name changes for the most part.

Team Meeting, 11Oct, 2009

Attendees:
Ryan Egger
Patrick Krepps
Marco Longoria
Patrick O'Neal

This meeting was devoted to refining user requirements, developing use cases, process models, and data flow diagrams.

Team Meeting, 18 Oct, 2009

Attendees:
Ryan Egger
Patrick Krepps
Marco Longoria
Patrick O'Neal

Due to the problems with dynamic location positioning updates, Patrick O'Neal has been exploring other options. Skyhook Wireless offers dynamic locationing applications, but the Wikitude API has no functionality that will access this information. Patrick discovered a front end application by the name of Gamaray that appears to allow dynamic positioning updates. Based on his recommendation, the team has decided to move away from Wikitude and to start developing in conjunction with Gamaray.

Testing of Gamaray required the device be upgraded to Android v1.5, which was successfully completed. Patrick and Marco then performed basic testing of Gamaray integration, which was also successful.

Ryan further refined several use cases.

The team agreed to meet again on Tuesday afternoon so we could begin object oriented analysis, and to begin developing class diagrams.

Team Meeting, 20 Oct, 2009

Attendees:
Ryan Egger
Patrick Krepps
Patrick O'Neal

The team did some basic class diagram analysis and further discussed the switch from Wikitude to Gamaray.

Team Meeting, 27 Oct, 2009

Attendees:
Ryan Egger
Patrick Krepps
Marco Longoria
Patrick O'Neal

The team has chosen to develop the project using the Gamaray engine. This meeting was devoted to continuing to learn the Gamaray system.

Ryan and Patrick O worked on further development and POI implementation. Marco and Patrick O worked on additional design details, and Ryan and Patrick K worked on further refining use cases, functional scenarios, and entity modeling.

Team Meeting, 01 Nov, 2009

Attendees:
Ryan Egger
Patrick Krepps
Patrick O'Neal

This meeting was mostly devoted to working on various aspects of the project.

Ryan developed a working prototype of the Options menu and the options, including the Tutorial and the Credits page.

Patrick O'Neal has continued to learn the Gamaray system, and is making great progress in implementing it. The problem he is running into is the vast amount of code the system uses that is unnecessary for our project. He has been trying to weed as much of it out as possible.
Patrick has also implemented a Subversion versioning system for code development.

Patrick Krepps has continued to develop Collaboration, Sequence, and class diagrams, and has continued to describe use cases, functional models, and data flow diagrams.

Team Meeting, 08 Nov, 2009

Attendees:
Ryan Egger
Marco Longoria
Patrick Krepps
Patrick O'Neal

The overwhelming amount of code involved in running Gamaray has become too much of a hindrance. It introduces a massive amount of unused code that must be loaded onto the device. As a result, earlier this week the team decided to drop the Gamaray development system in favor of completely custom code. Patrick O'Neal has already determined that it is possible to receive the necessary location, and other position/orientation information to make the system work.

Ryan has done a large amount of work on implementing the database into the application, and Marco has been doing the data entry. Patrick K has continued to develop documentation. At this point code development is outpacing that activity but the basic design remains the same. It is just a matter of making several name changes for the most part.

Team Meeting, 15 Nov, 2009

Attendees:
Ryan Egger
Marco Longoria
Patrick Krepps
Patrick O'Neal

Patrick O'Neal has made tremendous progress in coding the application, and Ryan has made as much progress implementing and integrating the database. Marco has completed the POI data entry, and has created basic web pages for the POIs that do not have their own web page. Patrick Krepps has been concentrating on the assignment from another class that he and Patrick O'Neal are teammates in, to allow Patrick O'Neal the needed time to work on this project. He has also continued to develop the project documentation.

Team Meeting, 16 Nov, 2009

Attendees:
Ryan Egger
Marco Longoria
Patrick Krepps
Patrick O'Neal

The team met in CCH-224 to finalize the Design presentation. We developed the presentation slides and documentation.

Team Meeting, 22 Nov, 2009

Attendees:
Ryan Egger
Marco Longoria
Patrick Krepps
Patrick O'Neal

The entire meeting was dedicated to finalizing product features and debugging/testing all aspects of the code.

Team Meeting, 29 Nov, 2009

Attendees:
Ryan Egger
Patrick Krepps

Patrick O'Neal has been excused due to illness. Ryan and Patrick Krepps began putting together the necessary elements for the final presentation due on December 3rd. The team will meet again on Tuesday and Wednesday to finalize the presentation.

Team Meeting, 06 Dec and 07 Dec, 2009

Attendees:
Ryan Egger
Patrick Krepps
Patrick O'Neal

The entire focus of both meetings was to gather all existing project documentation and components, to edit everything, and to produce the few remaining items. Several items were in "rough draft" format and had to be put into presentation form.

Team Meeting, 07 Dec, 2009

Attendees:
Ryan Egger
Patrick Krepps
Patrick O'Neal

The three of us are all working on creating a final report due on December 8th. This should be the final team meeting.

AR-Team 28 Sep 2009
Islander Tours Augmented Reality Application Status Report

Accomplishments

Met with Leticia Bazan, and obtained the Office of Student Recruitment and New Student Programs as project client
Completed project selection
Developed partial list of project requirements
Completed development platform selection
Completed OS and API selection
Installed all required development software and platform emulators on team computers

Planned Tasks

Team members to learn basic development skills (estimate: one to two weeks)
Refine functional requirements based on customer input (estimate: one week)
Refine non-functional requirements (estimate: one week)
Develop Use Case Analysis and Process Models (estimate: one to two weeks)

Open Issues

The team is getting adjusted to the new platform and API. All team members need to learn the basics of both the development system as well as the target platform. We all made progress in this area at the meeting on 27 Sep 2009.
The Wikitude API supports our intended design quite well, so our next big hurdle (the design phase) should be no problem.
We have received minimal feedback from the client, and promised materials have not been delivered. Patrick Krepps is attempting to contact an acquaintance that has acted as an Islander Ambassador in the past to solicit input from her.

AR-Team 10 Oct 2009
Islander Tours Augmented Reality Application Status Report

Accomplishments

Met with Laura Villalobos, former Islander Ambassador and Islander Tour tour guide.
Completed basic development tutorials. Everyone now has experience with the development system, the API, code generation, and transferring code to the development platform.
Developed project requirements to the best of our ability based on client input, and information received from Ms. Villalobos.
Developed a prototype of the final project. Due to the level of development familiarity at this point, it is uncertain if this is a throw away prototype or not.
Began Use Case Analysis and Process Models based on client and Ms. Villalobos' input.
Assigned primary tasks:

Code Development - Patrick O'Neal and Ryan Egger
Design – Marco Longoria
Documentation – Patrick Krepps

Planned Tasks

Complete requirement analysis based on Instructor input (estimate: one week).
Continue Use Case Analysis and Process Model development (estimate: one to two weeks).
Begin to formalize user documentation (estimate: user documentation will be an on-going task)
Develop formal use cases, process models and data flow diagrams (estimate: one week)

Open Issues

The chosen development system seems to take a GPS reading once when the application is first initiated. We plan to include a refresh button if that becomes necessary to allow the application to become fully mobile.
One possible solution to the above issue is the SkyHook positioning system. It appears to be applicable to the development system. After ensuring that it is, we will begin to incorporate that into our prototypes.
We have not heard anything back from the client. At this point we need to get direction from our advisor as to how to proceed. Most likely we will be tasked with using a proof of concept approach and basing our requirements on the feedback we have so far.

AR-Team 22 Oct 2009
Islander Tours Augmented Reality Application Status Report

Accomplishments

Dr. Mahdy has agreed to be our client for this project, and has directed us to develop the application based on the input we have received so far, and our idea of what the end user would need.
We completed the requirements and use case analysis.
Marco has created a large amount of POI information and integrated it with the application.
A dataset of POIs has been created with all pertinent information applied.
A set of POI icons has been developed as well.
Patrick O'Neal, Ryan and Marco have made advances in the overall application development.
Patrick K has a good draft of the user documentation.
Patrick O'Neal has researched the dynamic positioning problem, and it appears that we will not be able to perform dynamic updates to the user position using Wikitude. He has made the recommendation that the team move to a front-end application known as Gamaray. The team has chosen to move to Gamaray, as dynamic positioning is a must-have requirement.
Progress has been made on creating functional modeling, class extraction, class diagrams, entity modeling, and an application state chart.
A rapid prototype based on the Wikitude API has been developed and discarded due to location update issues.
A rough proof of concept has been developed using the Gamaray engine.

Planned Tasks

Complete object oriented analysis (estimate: one week)
Develop entity class models (estimate: one week)
Investigate integrating SkyHook to allow dynamic position updates (estimate one week)
Further prototype development using Gamaray

Open Issues

The team will need to prioritize learning how to integrate the requirements with the newly chosen development platform
Ryan is investigating the use of Skyhook Wireless for dynamic positioning updates
Time constraints

AR-Team 05 Nov 2009
Islander Tours Augmented Reality Application Status Report

Accomplishments

The team has successfully implemented the Gamaray API. This will allow the application to perform dynamic positioning.
Marco has begun converting the POI data to work with the change in code.
Patrick O'Neal has developed a POI database and is making great progress towards implementing it into the application.
Patrick O'Neal has implemented a software versioning server to allow code development to progress smoothly
Ryan has made significant progress in developing the options portion of the application.
Ryan has implemented the ability to launch the device web browser horizontally.
Ryan has integrated GoogleMaps into the application.
Patrick K has completed the use case analysis, the collaboration and sequence diagrams.

Planned Tasks

Marco is looking into the ability to provide dynamic content to the application database using SMS text messaging as an update mechanism (estimate: one week).
Take Process Modeling diagrams from rough draft to final form (estimate: one week).
Finalize functionality and code (estimate: one week).
Code testing and debugging (estimate: two weeks).

Open Issues

The following functionality will be implemented into the application, and most are now included in a rudimentary form at least:
Standard Islander Tour
Ability to apply POI filters to limit POIs to user's choice
The ability for the user to add a custom POI
The ability to search for particular POI information
An Options page that will contain:
An application tutorial, both text based and web based
An “About Us” page that will provide background information
A “Tell A Friend” link that will allow the user the ability to email application information to an email recipient
A link to GoogleMaps
Application development has outpaced formal documentation. This is not seen as a major issue as most code development has been discussed and agreed on by all team members. We just lack the formal documentation of the process.

AR-Team 17 Nov 2009
Islander Tours Augmented Reality Application Status Report

Accomplishments

The team has chosen to develop the application using custom code. We are unable to find an API that provides the necessary functionality, or that provides the functionality we need without using large quantities of the device resources.
Ryan has completed the application POI database. The database was de-normalized to save a bit on the amount of device memory required to implement it. This makes queries less efficient, but actually improves the application performance by limiting the amount of resources used.
Marco has completed the POI data entry into the database, and is creating a custom web page for each POI that does not already have an existing web page. This task is nearing completion
Patrick O'Neal has developed almost all of the project classes, methods, and data structures. He has developed a massive amount of code for the project, and has made a tremendous amount of progress towards implementing the functionality we required in the first place. The application has the ability to display POIs by category and by range. It has the ability to display web pages horizontally which allows the user to view the web page with the device orientated in the same direction as it is used by the application. The application updates the device location in near real-time, and computes the appropriate POI display based on all factors.
Patrick O'Neal's software versioning server has been a tremendous help.
Ryan is nearing completion of the options portion of the application.
Patrick K has completed the use case analysis, the collaboration and sequence diagrams. The team is finalizing the rest of the design documentation in preparation for the design specification presentation.
The team has presented the application idea to three separate computer science classes, and surveyed the students for possible enhancements to the application.

Planned Tasks

Finalize functionality and code (estimate: one week).
Code testing and debugging (estimate: one week).

Open Issues

The application is difficult to see in bright daylight.
At this point the application is nearing completion. The other main open issues that are still outstanding are the various enhancements we would like to implement, and the lack of time in which to do so.
If the POI does not have a web page, we really need to remove the links that go nowhere from the custom web page.

User Manual

IslanderAR

Texas A&M Corpus Christi
Augmented Reality Tours

Ryan Egger, Patrick Krepps, Marco Longoria, Patrick O'Neal

Table of Contents

Table of Contents 1
List of Illustrations 1
Chapter 1 What is IslanderAR 1
Chapter 2 Obtaining and Installing IslanderAR 2
Chapter 3 IslanderAR Basics 3
  Navigating the Home Screen 3
  IslanderAR Buttons 4
Chapter 4 Touring The Island University with IslanderAR 6
  Points of Interest 6
  Augmenting Reality 7
  Setting POI Display Distance 9
  Displaying POI Distance 10
  Enabling POI Distance Mode 10
Glossary 11

List of Illustrations

Figure 1 - IslanderAR Splash Screen 3
Figure 2 - IslanderAR Home Screen 4
Figure 3 – Information Sub-Menu 5
Figure 4 – Points of Interest Selection Screen 6
Figure 5 – Initial IslanderAR View 7
Figure 6– University Center Web Page 8
Figure 7– Setting the POI Distance Display 9
Figure 8 – The IslanderAR distance slider 9
Figure 9 – Displaying POI 10
Figure 10 – POI Range Mode Display 10

Chapter 1 What is IslanderAR

IslanderAR is an augmented reality (AR) application designed to run on cellular smart phones, devices that offer advanced functionality above that of a typical cellular phone. Smart phones have many of the capabilities of a modern Personal Computer (PC), including sophisticated operating systems with the ability to run multiple applications.

Augmented reality is a phrase that describes the merging of a view of the real, physical world with computer generated images. The computer generated images enhance, or augment, the reality view, giving more information than either system alone.

IslanderAR is designed to run on the Android™ Mobile Operating system. Android™ is a Linux based operating system that allows developers to write applications to control the device, using the Java language.

IslanderAR allows the user to conduct augmented tours of the Texas A&M University – Corpus Christi campus. The application works by transmitting the smart phone's camera view to the screen and overlaying icons representing specific campus sites known as Points of Interest, or POIs. As the camera is turned towards a POI, the icon for that POI floats into view in the same position as the POI.

IslanderAR takes real-time position measurements, and calculates which POIs are in the same direction as the camera view. It then causes these small icons to appear where the POI is. Note that IslanderAR will display all POIs in the camera direction, regardless of whether or not the user can actually see them. For instance, if one building is behind another, the user will see the icons for both. In this way the user that is unfamiliar with the campus can find items that they can not even see!

Current categories include information on Campus Buildings, Food Services, Parking Lots, Student Services, and Campus Landmarks.

IslanderAR is highly configurable in what it will display, making it quite easy for the user to find just the information they are interested in. IslanderAR is also user-friendly and easy to use.

Chapter 2 Obtaining and Installing IslanderAR

IslanderAR can be downloaded from the Android marketplace (http://www.android.com/market/).

If you wish to see and use the source code itself, you will need a computer on which to develop, and the ability to view/modify java source code. For detailed instructions, see the “IslanderAR development guide”.

Chapter 3 IslanderAR Basics

Navigating the Home Screen

When IslanderAR is first launched, the application must initialize several variables and settings, and it must initiate Global Positioning System (GPS) communication with the GPS satellite system. The user is presented with the IslanderAR splash screen that displays while the application is initializing. See Figure 1.

Figure 1 - IslanderAR Splash Screen

Once IslanderAR has finished initializing, the user is presented with the Home Screen, from which all activity is begins. The Home Screen contains three menu buttons. They are **Points of Interest**, **Search Points of Interest**, and **Options**. See Figure 2.

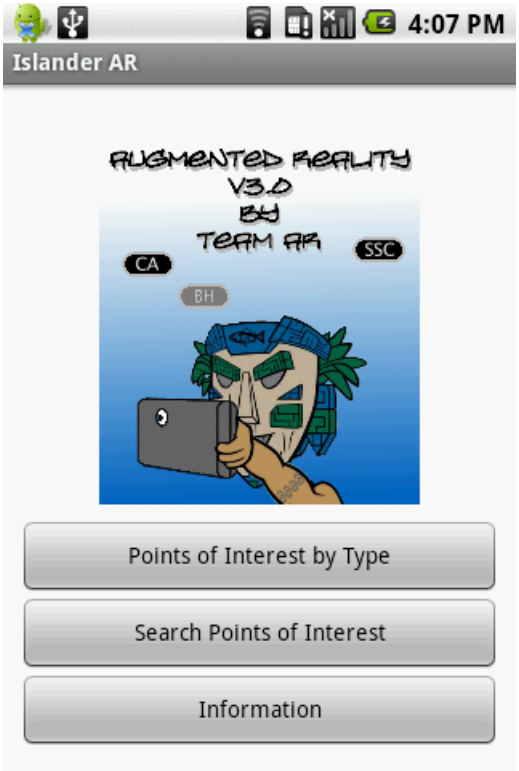


Figure 2 - IslanderAR Home Screen

IslanderAR Buttons

The **Points of Interest** button launches the IslanderAR tour and allows the user to select the types of Points of Interest (POIs) they would like to obtain information on. Current categories are: Campus Buildings, Food Services, Parking Lots, Student Services, and Campus Landmarks.

The **Search Points of Interest** button allows the user to search for a specific POI, such as a particular building or student service. If the POI is not within the camera direction, an arrow will be displayed on one side of the view indicating which way the user should turn to bring the POI into view. As the POI comes into camera view, the arrow will be replaced with the system icon for the POI.

The **Information** button presents the user with a sub-menu. Selections on the Information List sub-menu are **What is a Point of Interest**, **About**, **Tutorial**, **Tell a Friend**, and **Acknowledgments**. See Figure 3.

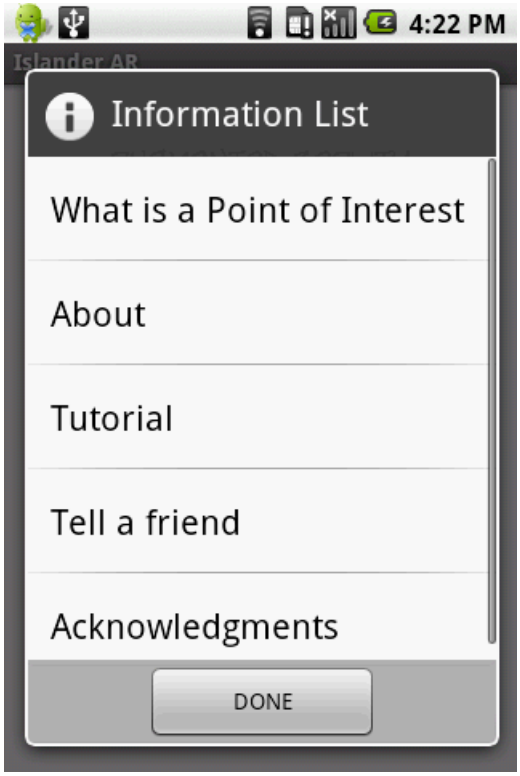


Figure 3 – Information Sub-Menu

Two of the buttons display brief, static information. The **What is a Point of Interest** button will display an information dialog describing what a Point of Interest is. The **About** button will display an information dialog describing the IslanderAR application.

The remaining three buttons provide more dynamic content. The **Tutorial** button will direct the device's web browser to a basic web-based tutorial. The tutorial will teach the user about IslanderAR. The **Tell A Friend** button will launch an email client so the user can tell a friend about IslanderAR using the device's email client (a valid data plan may be required to use this function). The **Acknowledgement** button will launch the device's web browser and load a site that contains information about the people that developed IslanderAR.

The main menu can always be returned to from any of the sub-menus by pressing the device's back button. As the user traverses the IslanderAR menus, the previous menu can be returned to by pressing the back button until the user has returned to the main menu.

Chapter 4 Touring The Island University with IslanderAR

Points of Interest

Pressing the **Points of Interest** button presents the user with a Points of Interest menu. The menu has five selections available, with checkboxes next to each. The user must select at least one category, and may select as many as all five. See Figure 4.

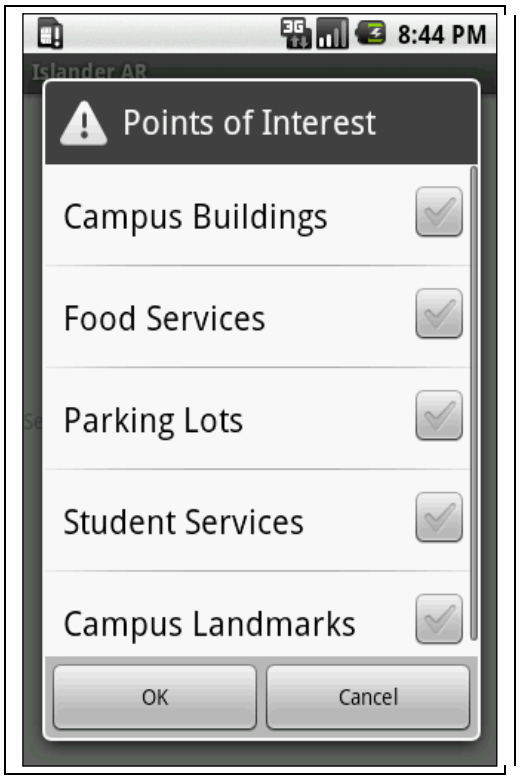


Figure 4 – Points of Interest Selection Screen

Once the user has made a selection, they can then click OK to initiate the augmented reality view, or Cancel to return to the main menu. If they click OK, the application launches the augmented reality view in a horizontal viewing mode. See Figure 5.

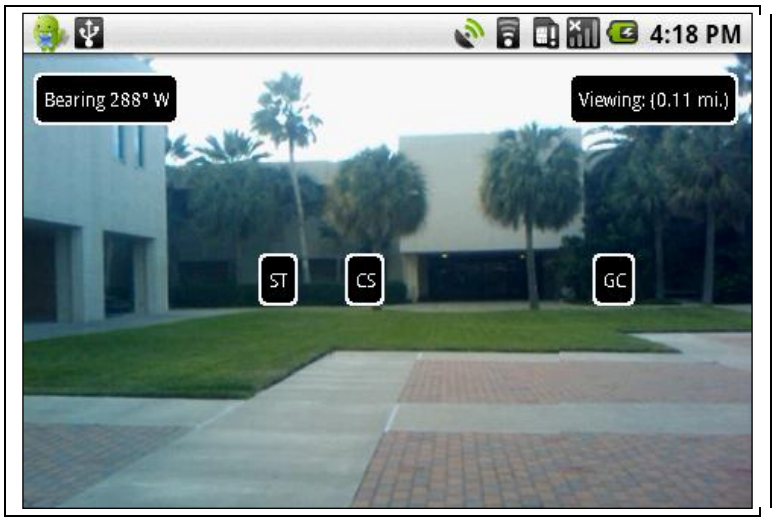


Figure 5 – Initial IslanderAR View

As you can see, IslanderAR displays the camera view with several POI icons in the center of the screen. As you move the phone, the icons will shift to remain in positions relative to their location and your position. As you move about, POI icons will enter and leave the screen as the distance to them changes.

In the top left corner of the display is the device's current bearing. This is the compass heading of the direction the camera is currently pointing towards. In the top right corner of the screen is the current distance setting. Only POIs within this distance will be visible. The distance can be configured dynamically, and instructions for doing so are in the Setting POI Display Distance section.

Augmenting Reality

IslanderAR does more than simply display POI icons on the screen. Clicking on an icon will cause a textbox to appear that contains a brief description of the POI. Clicking the Back button will return the user to the POI view, but clicking the More Info button brings a wealth of information to the user.

The More Info button launches the IslanderAR web browser and takes the user to a website that offers more detailed information about the POI. From this web site the user has links to other information, and any existing POI web page.

For instance, clicking the UC icon when it is in view, and then clicking on the More Info button will launch the IslanderAR web page that contains concise information about the UC, and a link to the University Center web page that contains links to the many student services and merchants that reside in the UC. See Figure 6

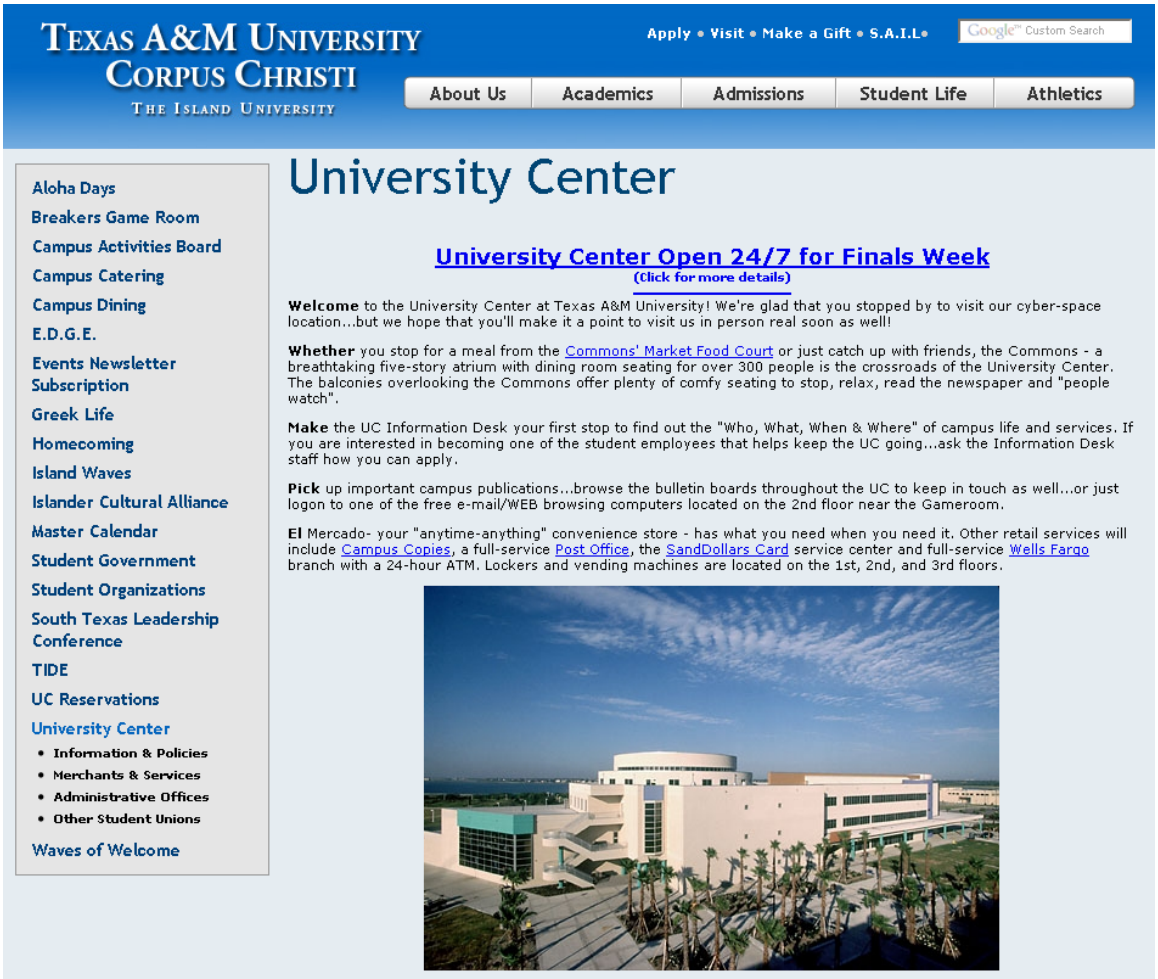

TEXAS A&M UNIVERSITY
CORPUS CHRISTI
THE ISLAND UNIVERSITY
Apply • Visit • Make a Gift • S.A.I.L•
Google™ Custom Search
About Us
Academics
Admissions
Student Life
Athletics
Aloha Days
Breakers Game Room
Campus Activities Board
Campus Catering
Campus Dining
E.D.G.E.
Events Newsletter Subscription
Greek Life
Homecoming
Island Waves
Islander Cultural Alliance
Master Calendar
Student Government
Student Organizations
South Texas Leadership Conference
TIDE
UC Reservations
University Center
Information & Policies
Merchants & Services
Administrative Offices
Other Student Unions
Waves of Welcome
University Center
University Center Open 24/7 for Finals Week
(Click for more details)
Welcome to the University Center at Texas A&M University! We're glad that you stopped by to visit our cyber-space location...but we hope that you'll make it a point to visit us in person real soon as well!
Whether you stop for a meal from the Commons' Market Food Court or just catch up with friends, the Commons - a breathtaking five-story atrium with dining room seating for over 300 people is the crossroads of the University Center. The balconies overlooking the Commons offer plenty of comfy seating to stop, relax, read the newspaper and "people watch".
Make the UC Information Desk your first stop to find out the "Who, What, When & Where" of campus life and services. If you are interested in becoming one of the student employees that helps keep the UC going...ask the Information Desk staff how you can apply.
Pick up important campus publications...browse the bulletin boards throughout the UC to keep in touch as well...or just logon to one of the free e-mail/WEB browsing computers located on the 2nd floor near the Gameroom.
El Mercado- your "anytime-anything" convenience store - has what you need when you need it. Other retail services will include Campus Copies, a full-service Post Office, the SandDollars Card service center and full-service Wells Fargo branch with a 24-hour ATM. Lockers and vending machines are located on the 1st, 2nd, and 3rd floors.

Figure 6– University Center Web Page

The device's web browser is a full-featured web browser, and all of the links on the web page brought up by the IslanderAR browser will lead to additional pages. If the user clicks on a link that launches the device's web browser, the IslanderAR process is put to sleep until the device's back button is pressed. At that point the web browser is closed and IslanderAR wakes up and continues displaying augmented reality.

Setting POI Display Distance

In the **Points of Interest** section, it was noted that IslanderAR displays only POI icons for POIs that are within a certain distance of the user. By default, only POIs within 0.1 miles are displayed. This distance can be changed, from a minimum of 0 miles to a maximum of 1 mile.

To set the POI display distance, press the device's Menu button to bring up the IslanderAR menu. The IslanderAR menu consists of three buttons along the bottom of the screen, **Set Distance**, **Show POI Distance**, and **Enable Distance Mode**. See Figure 7

Figure 7– Setting the POI Distance Display

Selecting the **Set Distance** button will display a distance slider control. The user can drag the slider to the desired distance and select OK to set the distance. Only POIs within that distance are then displayed. See Figure 8.

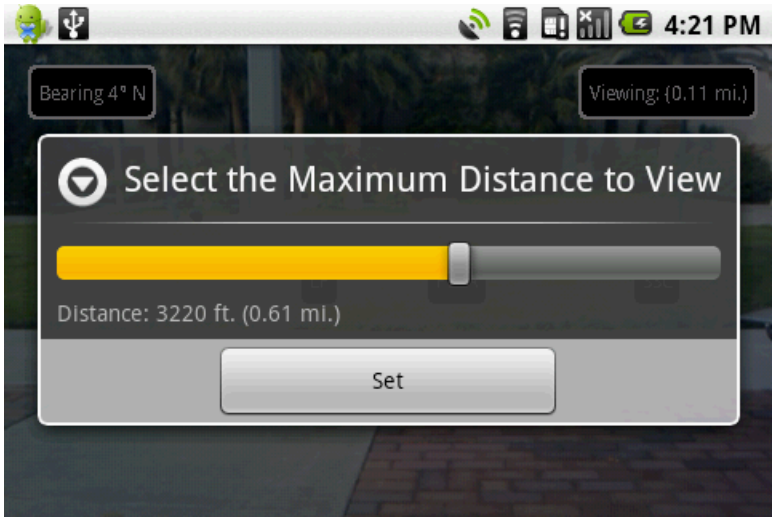


Figure 8 – The IslanderAR distance slider

Displaying POI Distance

Selecting the **Show POI Distance** button will cause the distance from the user to the POI to be displayed in the POI icon along with the POI name. See Figure 9.

Figure 9 – Displaying POI

Enabling POI Distance Mode

In addition to setting the POI display distance, the POI view can be changed to indicate distance in one more way. From the IslanderAR menu, the user can invoke the IslanderAR POI distance mode. In this mode, POIs are displayed in a three-tiered manner, with the closest POIs on the bottom tier, the mid-distance POIs on the center tier, and the furthest POIs on the top tier. In addition to layering the POIs in a tiered manner, the background color of the POI icon is solid black for the closest POIs, and light grey for the most distant. The mid-distance POI icons are colored medium gray. This gives the augmented view a quasi three-dimensional appearance. See Figure 10.

Figure 10 – POI Range Mode Display

Glossary

Augmented Reality – Merged view of the real world with technologically derived information to provide additional information beyond normal reality.

Bearing – The compass direction in which someone is facing.

GPS – **G**lobal **P**ositioning **S**ystem. A system of determining one's absolute position on the Earth's surface by obtaining accurate timing signals from satellites in precisely known orbits.

POI – Point of Interest. An existing structure or service that a person would have an interest in learning more about than they already know

Smart Phone – A handheld mobile device that provides all the functionality of a cellular phone as well as PC-like functionality.

Source Code Installation Guide

Source Code Installation Guide

In order to install and execute the Android source code for the IslanderAR capstone project, the following steps must be taken.

Please note that in this installation guide the steps to download and install the Android SDK version 1.5 are given. This is done intentionally, as the application was developed and tested using the Android SDK v1.5. We do not make any guarantees as to the accuracy or completeness of the application using any other version of the SDK, namely v1.6 or v2.0. Theoretically, importing this application into newer versions of the SDK should work. However, we have not tested them, and therefore do not recommend anything other than v1.5.

1. **Installation Requirements**
   - 1.1. Before installation, please ensure that you have the current version of Java installed on your machine.
     - 1.1.1. This can be accomplished by visiting http://java.com/en/
     - 1.1.2. Select the "Do I have Java?" link.
     - 1.1.3. Follow on screen instructions if you do not have the current version.

2. **Download and Install the Eclipse 3.5 IDE**
   - 2.1. Start a web browser, and navigate to http://www.eclipse.org/downloads/
   - 2.2. Identify the package of Eclipse labeled "Eclipse IDE for Java Developers"
   - 2.3. For said package, select the appropriate operating system download that best fits your platform. (i.e. Windows/Mac/Linux)
   - 2.4. Download the selected .zip archive to a directory of your choosing.
   - 2.5. Extract the .zip archive.
   - 2.6. Once the .zip archive is extracted, navigate to the "eclipse" directory.
   - 2.7. Select and execute the "eclipse.exe" application.
   - 2.8. Upon successful startup, Eclipse will prompt you to "Select a workspace."
     - 2.8.1. Select any directory of your choosing.
     - 2.8.2. Click the **OK** button to continue

3. **Download the Android Software Development Kit**
   - 3.1. Start a web browser and navigate to http://developer.android.com/sdk/index.html
   - 3.2. Identify and download the appropriate package, given your current operating system platform. (i.e. Windows/Mac/Linux)
     - 3.2.1. Note: You will need to agree to a terms of use document to continue.
   - 3.3. Download the chosen .zip archive to a directory of your choosing.
   - 3.4. Extract the contents of the "android-sdk" archive to directory that seems appropriate. Ensure that you remember the location for future reference.
   - 3.5. Navigate to the directory in which you extracted the contents of the .zip archive.
   - 3.6. Select and execute the "SDK Setup" application
   - 3.7. Upon launching the "SDK Setup" application, you should be presented with a "Choose Packages to Install" prompt.
     - 3.7.1. **Note**: If you are presented with a "Refresh Sources" prompt that displays a message containing the phrase "Failed to fetch URL…"
     - 3.7.2. Click the **Close** button on said prompt.
     - 3.7.3. Click the **Cancel** button on the "Choose Packages to Install" prompt.

3.7.4. Select the **Settings** option in the list box on the left of the application.
3.7.5. **Check** the box for "Force https:// … sources to be fetched using http://…"
3.7.6. Click **Save & Apply**
3.7.7. Restart the application.
3.8. From the "Choose Packages to Install" prompt, select **SDK Platform Android 1.5 API3, revision 3**
3.9. Ensure that the **Accept** radio button is selected.
3.10. Click **Install Accepted**
3.11. The download and installation of the SDK has begun. Take a break, and grab a cup of coffee.
3.12. Once the message "Installed SDK Platform Android…" appears, click **Close**, and then exit the application.

4. Download and Install the Android Development Tools (ADT) plugin for Eclipse IDE

4.1. Start Eclipse; select **Help → Install New Software** from the main menu bar at the top of the application's screen.
4.2. In the field labeled "Work with:" enter https://dl-ssl.google.com/android/eclipse/ and press the **Enter** key.
4.3. Give the application a moment to search the aforementioned link for Eclipse plugins. Once the "Pending" message disappears, you should be presented with a line that reads "Developer Tools."
4.4. **Check** the box next to "Developer Tools."
4.5. Click the **Next** button.
4.6. Click the **Next** button on the "Install Details" page.
4.7. Review the terms of the license agreements, and then select the **I accept the terms…** radio button.
4.8. Click the **Finish** button.
4.9. The installation of the ADT plugin has begun.
4.10. Restart Eclipse when the installation is complete.
4.11. After Eclipse has restarted, select **Window → Preferences** from the main menu bar at the top of the application's screen.
4.12. Select **Android** from the list box on the left of the screen.
4.13. For the label that reads "SDK Location," click the **Browse** button.
4.14. At the directory tree view that launches, located and select the root directory of the Android SDK you extracted in Step 3.4. (For example "android-sdk-windows" if you downloaded the Windows version of the SDK.)
4.15. Click the **OK** button.
4.16. Click the **Apply** button on the preferences screen.
4.17. The Android Development Tools plugin is now successfully installed, and linked to the Android SDK.

5. Importing and Building the IslanderAR project from Source Code

5.1. Start Eclipse; select **File → Import** from the main menu bar at the top of the application's screen.
5.2. Select and expand the "General" folder from the "Import" prompt.
5.3. Select "Existing Projects into Workspace" from the "General" folder.
5.4. Click the **Next** button.
5.5. For the label that reads "Select root directory" click the **Browse** button.
5.6. Select the "IslanderAR" directory from the supplied CD-ROM.
5.7. Click the **OK** button.

5.8. **Check** the box next to "Copy projects into workspace"
5.9. Click the **Finish** button.
5.10. The IslanderAR project is now imported and built in your local workspace.

6. Executing the IslanderAR Application

There are two specific ways to execute Android applications. The first method requires that you attach an external development device and launch the given application via USB. This requires that the correct USB drivers are installed for the device in question. This method will not be covered in this guide, as drivers vary between the devices used. The second method involves launching the application on the virtual device emulator. Instructions on how to complete this task are enumerated below.

6.1. Create Android Virtual Device (AVD)
  6.1.1. Start Eclipse; select **Window → Android SDK and AVD Manager** from the main menu bar at the top of the application's screen.
  6.1.2. Ensure that you are viewing the "Virtual Devices" page as selected in the list box at the left of the screen.
  6.1.3. On the right side of the screen, click the **New** button.
  6.1.4. Assign a name of your choosing in the "Name:" field.
  6.1.5. Select "Android 1.5 – API Level 3" from the "Target" drop down box.
  6.1.6. Assign a value of your choosing into the "SD Card:" Size field. (16 is a good default.)
  6.1.7. Click the **Create AVD** button.
6.2. Running the IslanderAR Application
  6.2.1. In the "Package Explorer" of the main Eclipse "Java" perspective.
  6.2.2. Select the IslanderAR project.
  6.2.3. Select **Run → Run** from the main menu bar at the top of the Eclipse window.
  6.2.4. At the "Run As" prompt, select "Android Application"
  6.2.5. Click the **OK** button.
  6.2.6. The Android Virtual Device Emulator should now launch. This can be a time consuming process, depending on the resources of your machine. Please be aware of this fact.

Known Issues

Known issues:
The device GPS listener requires a clear view of the sky. It will acquire a position reading on cloudy days or inside buildings sometimes, but the margin of error in the reading becomes substantial.

The device compass, accelerometers, and GPS sensors are sensitive to magnetic and electrical interference at close range to strong electrical or magnetic fields.

The device sensors can become un-calibrated. The only known way to re-calibrate the sensors is to shake the device in a figure 8 motion.

An unknown math error in the position calculation causes the icons to disappear when the bearing transits from 359 to 0 or from 0 to 359 degrees.

The search feature requires exact input. If not given exact input the query returns nothing.

Recommended Future Work

There are some enhancements and some additional functionality we would have liked to have implemented.

Point of Interest (POI) database enhancements:
First and foremost would be the ability to dynamically update the POI database. Currently all POIs must be entered into the database by hand, and the application must then be re-compiled. The new database must then be added to the project, and the entire project must then be uploaded to the device.

We explored a couple of ways to implement dynamic updating of the database. One possibility would be to link to a database hosted on a separate server. This would obviously require a wireless connection or a data plan and a Global System for Mobile communications (GSM) connection. Another option discussed would be to host the database on a separate server and develop the ability to store a local copy on the device. This would alleviate the complete loss of functionality in the event of signal loss.

Another more esoteric idea that was discussed was the ability to update the local database on the device through the sending of Short Message Service (SMS) text messaging. This would be especially useful for listing current events dynamically. In the case of the IslanderAR application, events such as Island Days or sporting events would be a good example of this type of content.

In the case of dynamic content, we discussed adding the content with valid dates, so the POI for the event would not be displayed prior to a certain day, and would cease being displayed after the event had passed.

Planar Display Mode:
One other enhancement would be the ability to display the POI icons in such a way that if the user turned the device so that it was parallel to the ground, the POIs would still display in an intuitive manner. For instance the icon could change to a pentagon type shape with the pointed end pointing towards the POI. The search feature would still display an arrow that indicated the direction to turn, but it could also indicate the direction of the POI if the distance mode was set to mask the POI.

Icons:
Another idea discussed was the ability to have different categories of POI display different icons. For instance, eating establishments could be represented by an icon that looked like a plate and dinnerware place setting. An office building could be represented by an icon that looked like a high-rise office tower, and medical services could be represented by an icon that looks like the Staff of Asclepius.

Another icon enhancement that we discussed was to have the icons grow in size in relation to their relative distance. The user could set the distance range, and POI icons beyond that would not display. Icons just inside that range would be tiny dots, and they would grow into full size icons as the distance decreased to a preset minimum.

Breadcrumb:
Another feature that we wanted to implement was what we tentatively called a Breadcrumb. This would be the ability for the user to mark a specific spot and be able to return to it at any time. The most likely usage of this feature would be to mark a parking spot so the user could return to their car.

Positioning:
Another area of work that needs attention is user positioning. The Global Positioning System (GPS) positioning system is great when the user is outdoors on a clear day. However the margin of error increases substantially indoors or during inclement weather. We explored the use of Skyhook™ positioning system. Unfortunately, it offered no better solution for the campus as the campus apparently uses one Service Set identifier (SSID) for the entire campus. This prevents Skyhook from providing any better positioning than the current GPS system provides.

Enhanced positioning could be achieved with the use of some type of sensor system. We discussed the possibility of small Bluetooth transmitters placed at known locations. This would provide reliable, absolute positioning, and make it possible to create POIs for individual offices or rooms throughout the campus.

Possible future clients

The Chamber of Commerce – Business travelers could download the application when they check into their hotel. They would then have access to local restaurants, activities, merchants, and entertainment.

Realtors – Prospective home buyers could tour neighborhoods they like and pull up all local listings. The POI could provide details about any property, and launch virtual tours of the property using the device web browser.

City Parks – Park goers could find the nearest park with the features they want, or could obtain information about parks they are interested in.

Mall Directories – Shoppers could download the mall directory as they enter the mall. Not only would they have an up-to-date listing of all mall retailers, they could get sales and inventory information.

Golf courses – The application could direct golfers from hole to hole and back to the clubhouse.

Historical tours – Cities that conduct historical home tours could make the application available for people interested in the tours. Another example would be something along the lines of the Rockport Hummingbird Festival or the holiday Tour of Lights in some cities.

Museums – With better positioning, such as that possible with Bluetooth sensors, museum visitors could conduct their own self-paced guided tours. The application could even be enhanced with audio capabilities to augment the tour. Art galleries, historical sites such as battlefields, and major sporting events could also benefit from this as well.

Bibliography

Schach, Stephen R. *Object-Oriented & Classical Software Engineering,*. 7th ed. New York: Mc Graw Hill, 2007. Print.

"Android Development Reference." *Android Developers*. Web. 06 Dec. 2009. <http://developer.android.com/index.html>.

"Documentation." *SQLite Home Page*. Web. 06 Dec. 2009. <http://www.sqlite.org/>.

Haseman, Chris. "Augmented Reality on Android: Prepping the Camera and Compass. " DexX. 14 Aug. 2009. Web. 20 Oct. 2009. <http://www.devx.com/wireless/Article/42482>.

Haseman, Chris. "Augmented Reality on Android: Using GPS and the Accelerometer." DevX. 10 Oct. 2009. Web. 20 Oct. 2009. <http://www.devx.com/wireless/Article/43005>.

Murphy, Mark L. *Beginning Android*. 1st ed. New York: Apress, 2009. Print.

Rogers, Rick. *Android Application Development*. 1st ed. Sebastopol: O'Reilly Books, 2009. Print.

Rob, Peter. *Database Systems Design, Implementation, and Management*. 8th ed. Boston: Course Technology Cengage Learning, 2009. Print.